

# Model Compression for Efficient AI Computing

From TinyML to Large Language Model and GenAI



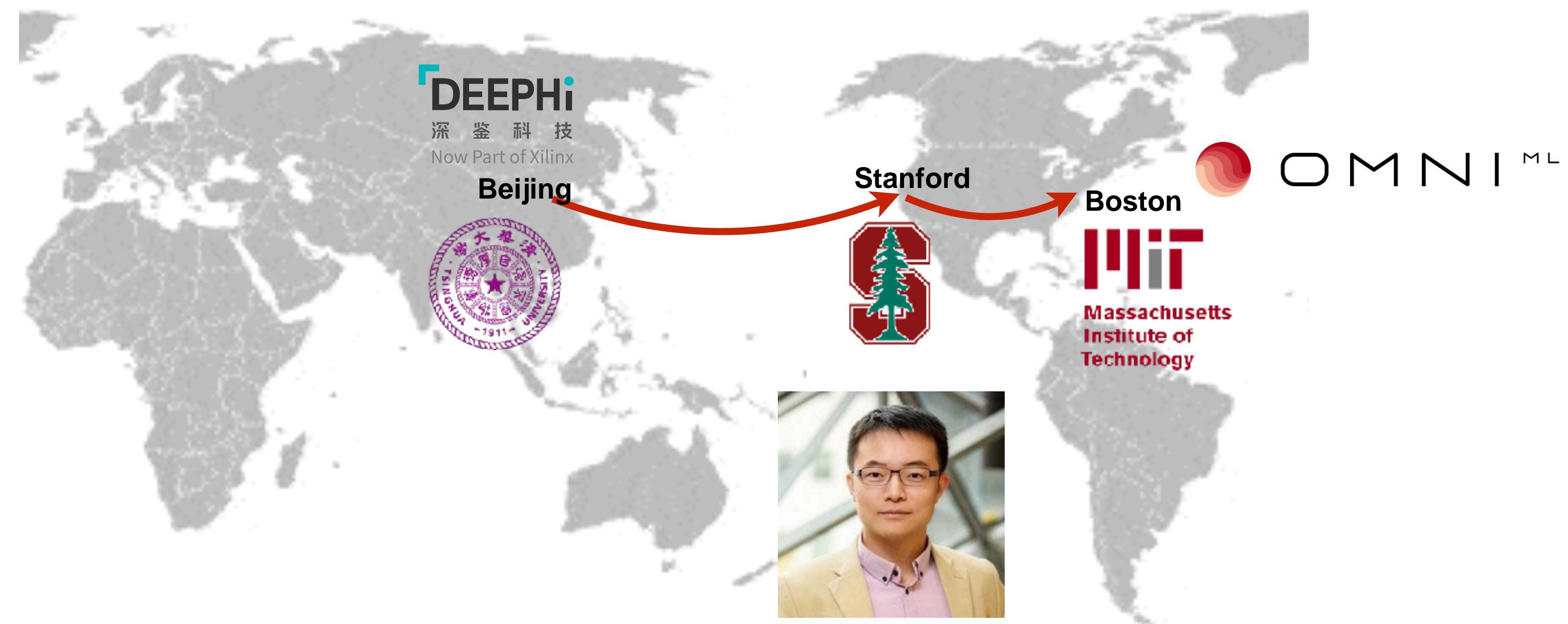
**Song Han**

Associate Professor, MIT  
Distinguished Scientist, NVIDIA  
<https://songhan.mit.edu>

 @SongHan/MIT

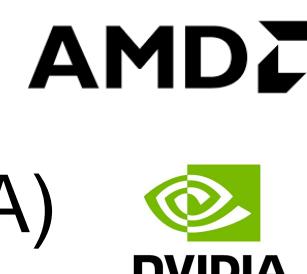


# Prof. Song Han



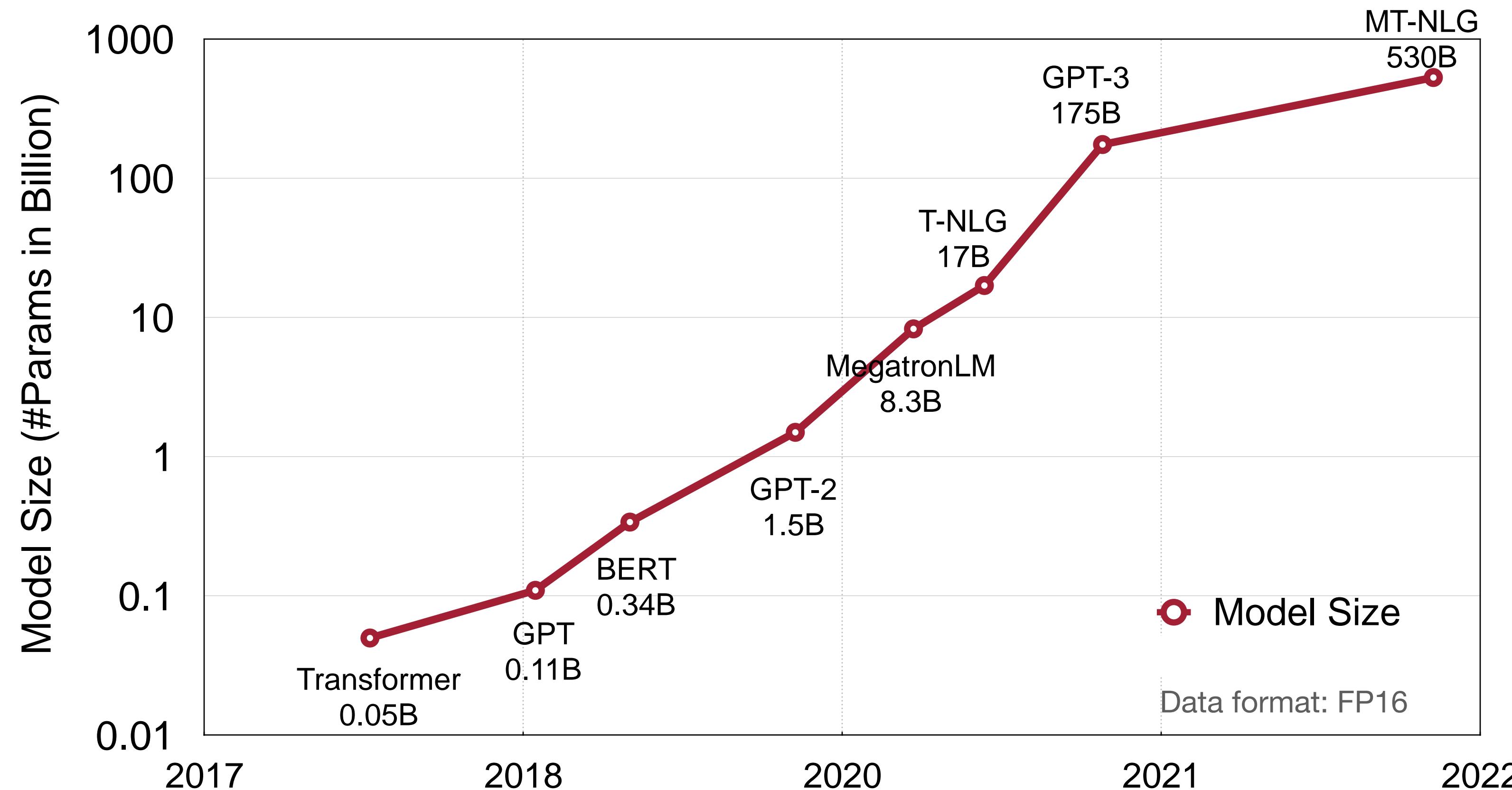
- B.S. from Tsinghua University
- Ph.D. from Stanford University, advised by Prof. Bill Dally
- Deep Compression (best paper award of ICLR)
- EIE (top5 cited paper in 50 years of ISCA)
- MIT Technology Review, 35 Innovators under 35
- NSF Career Award
- IEEE "AIs 10 to Watch: The Future of AI" Award
- Sloan Research Fellowship

- Cofounder of DeePhi (now part of AMD)
- Cofounder of OmniML (now part of NVIDIA)



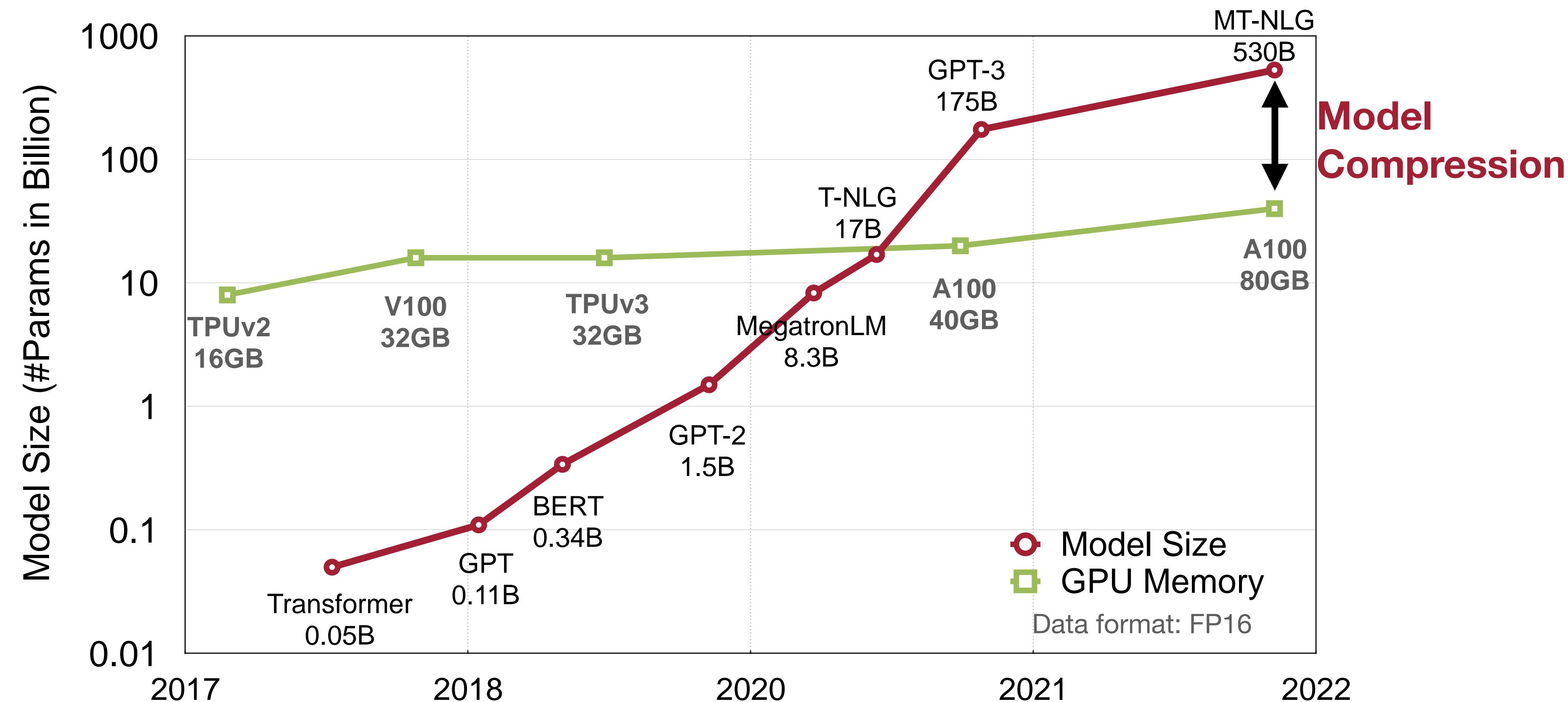
# Deep Learning Continues to Scale

The demand of computation grows exponentially



# Model Compression and Efficient AI are Essential

Bridge the Gap between the Supply and Demand of AI Computing



# Problem: DL Models Outgrow Hardware

Moore's Law: 2x transistors every 2 years;  
DL models: 4x parameters every 2 years



208V, 50Amp:  
2x NVIDIA DGX A100  
or  
1x NVIDIA DGX H100  
15KW!

# Model Compression Bridges the Gap

We need Green AI



**before**

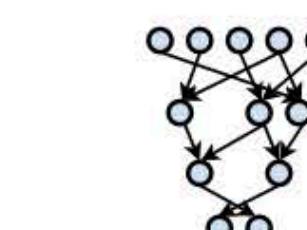
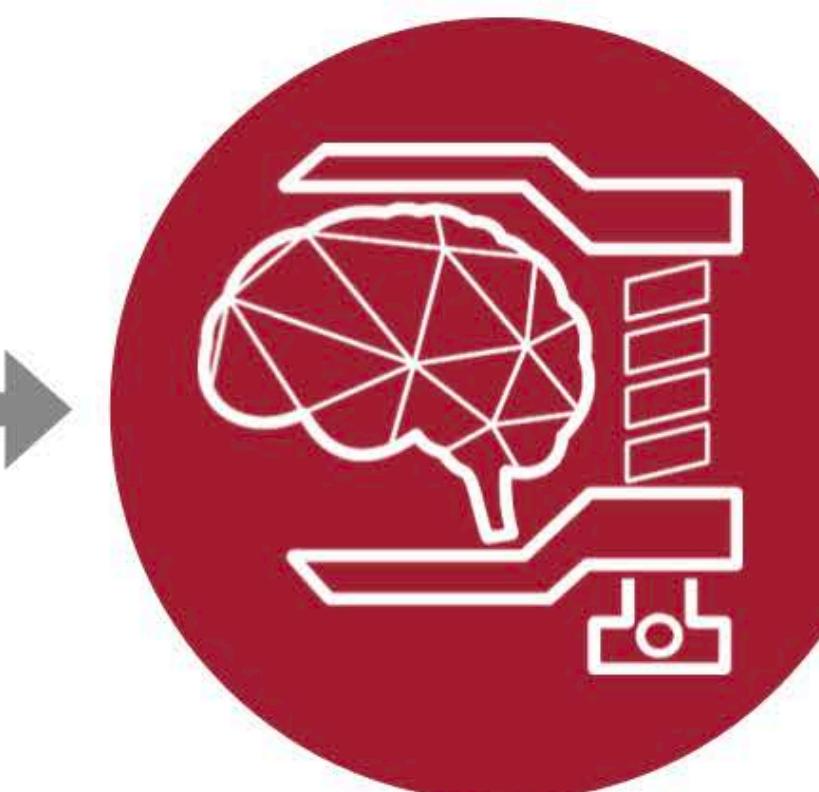
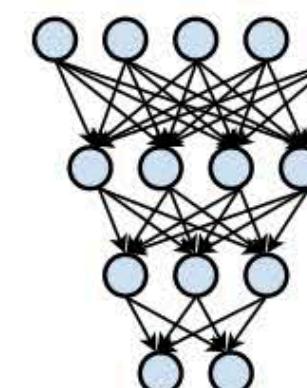
Training



Inference

**after**

Training

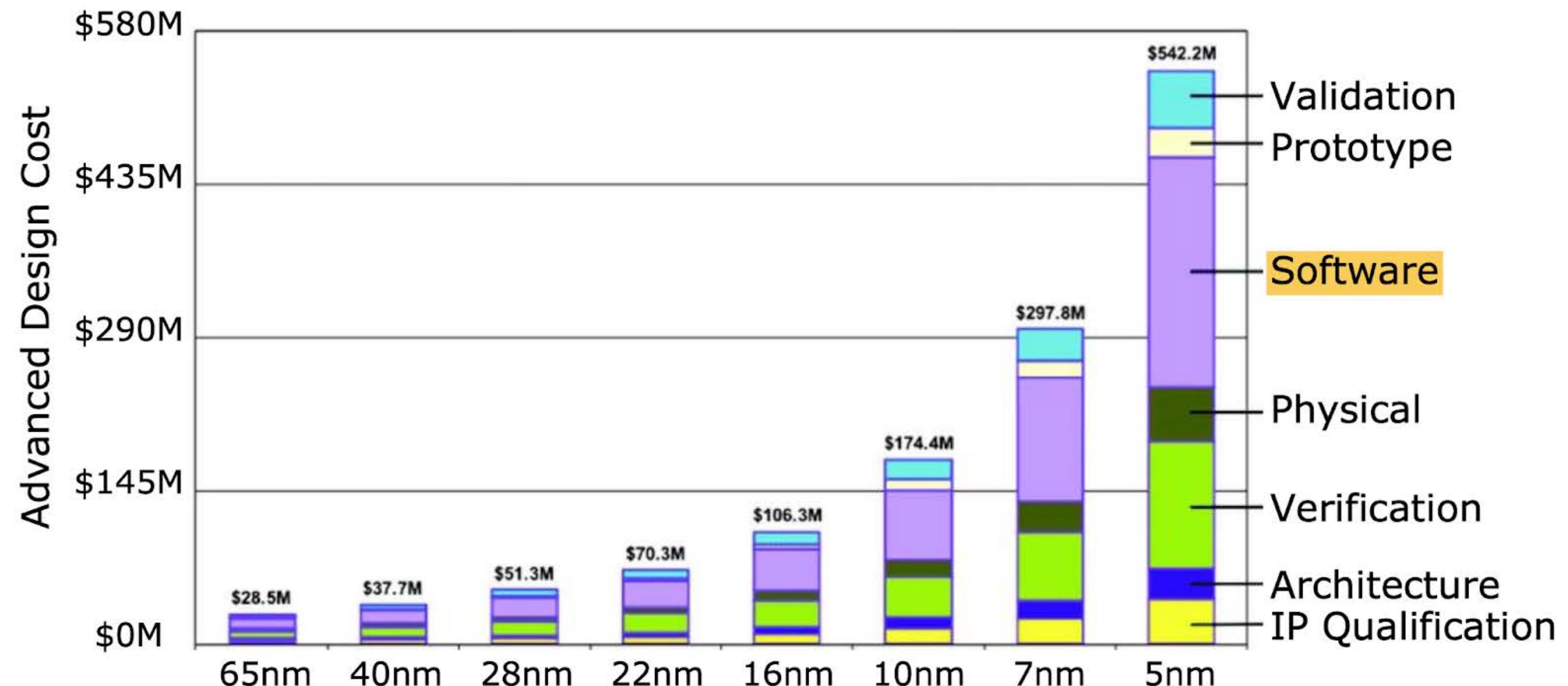


Inference

**Model compression:**  
Pruning, sparsity, quantization, etc

“Deep Compression” and EIE brings new opportunity to build hardware accelerator for sparse and compressed neural networks

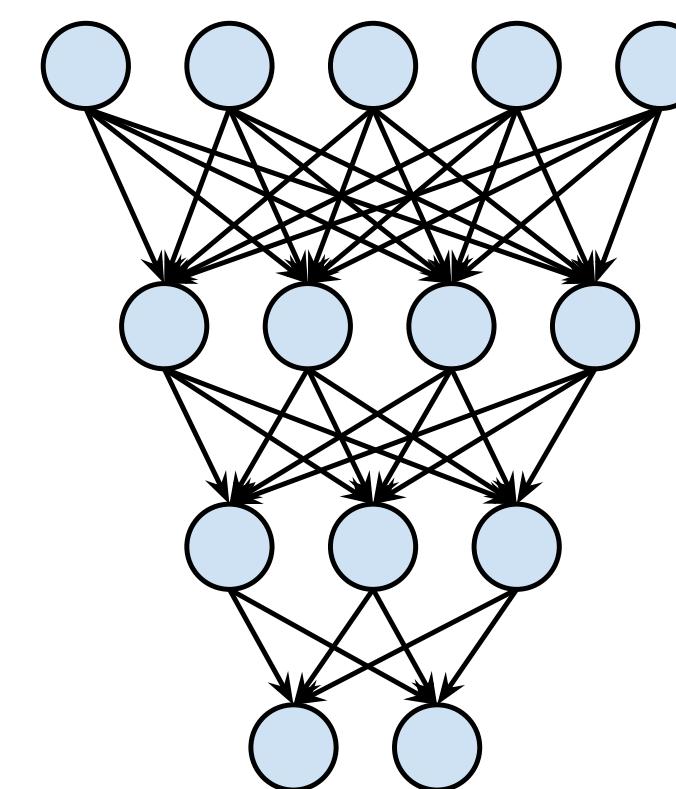
# Software is important in advanced technology node



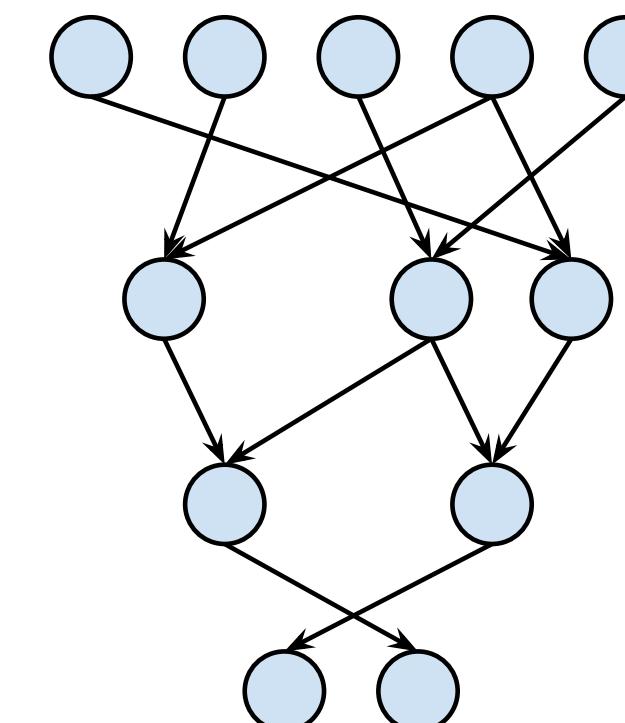
The software cost dominates the cost breakdown of advanced technology nodes [\[source\]](#).

We focus on designing new algorithms and software for efficient computing.

# Deep Compression



Pruning



weights (32 bit float)			
2.09	-0.98	1.48	0.09
0.05	-0.14	-1.08	2.12
-0.91	1.92	0	-1.03
1.87	0	1.53	1.49

cluster

Quantization

cluster index (2 bit uint)			
3	0	2	1
1	1	0	3
0	3	1	0
3	1	2	2

centroids
3: 2.00
2: 1.50
1: 0.00
0: -1.00

Original ResNet-50



100MB

with Deep Compression

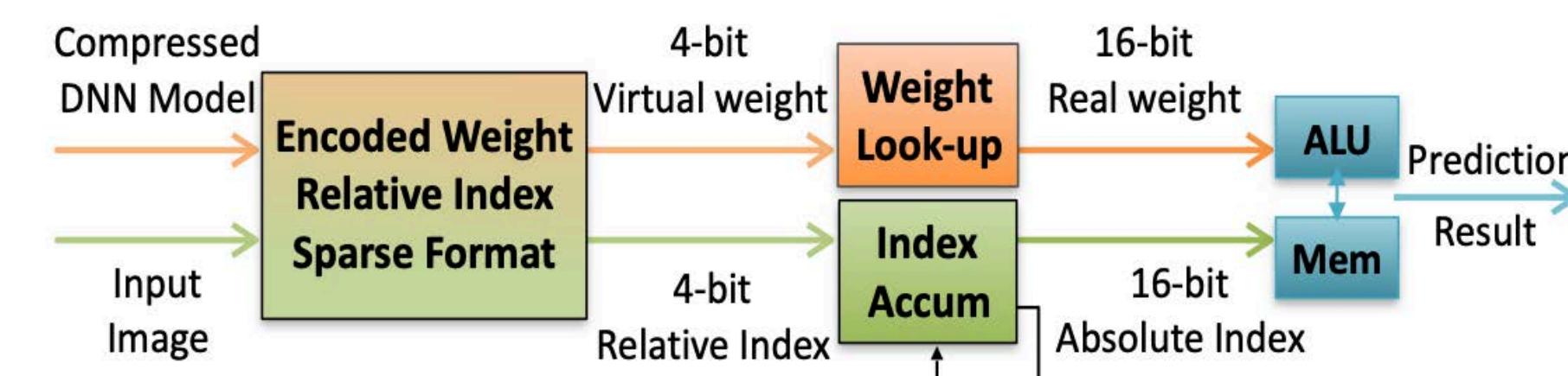


6MB

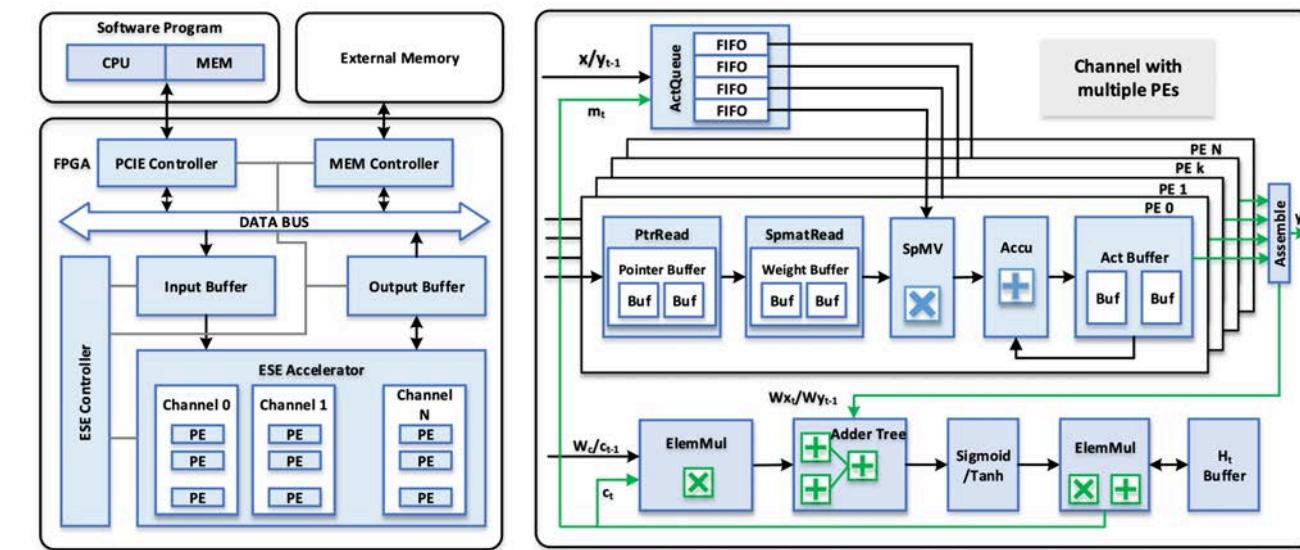
17x compression

# Hardware Support for Sparsity

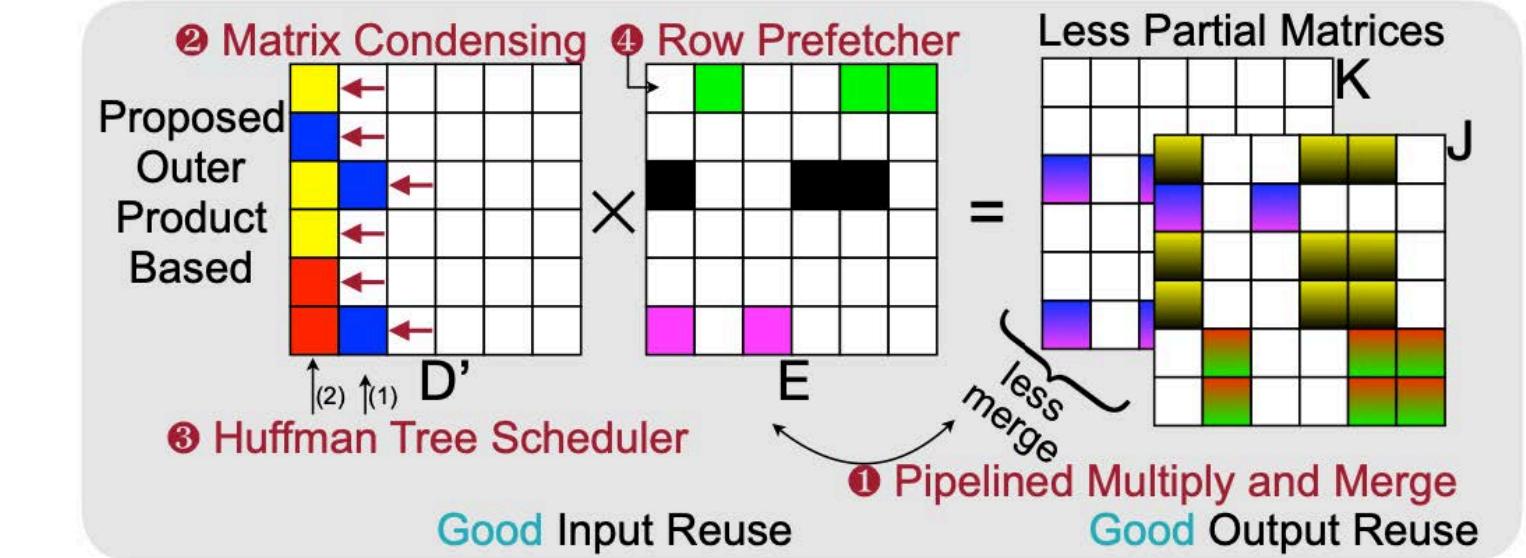
EIE (Efficient Inference Engine) brings weight sparsity to AI accelerators



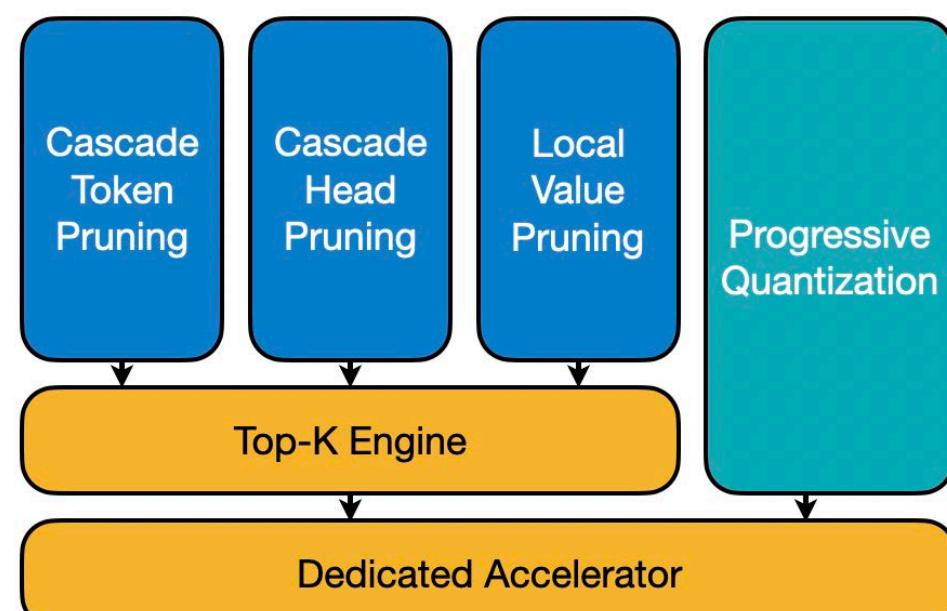
[EIE](#), Han et al, ISCA'16



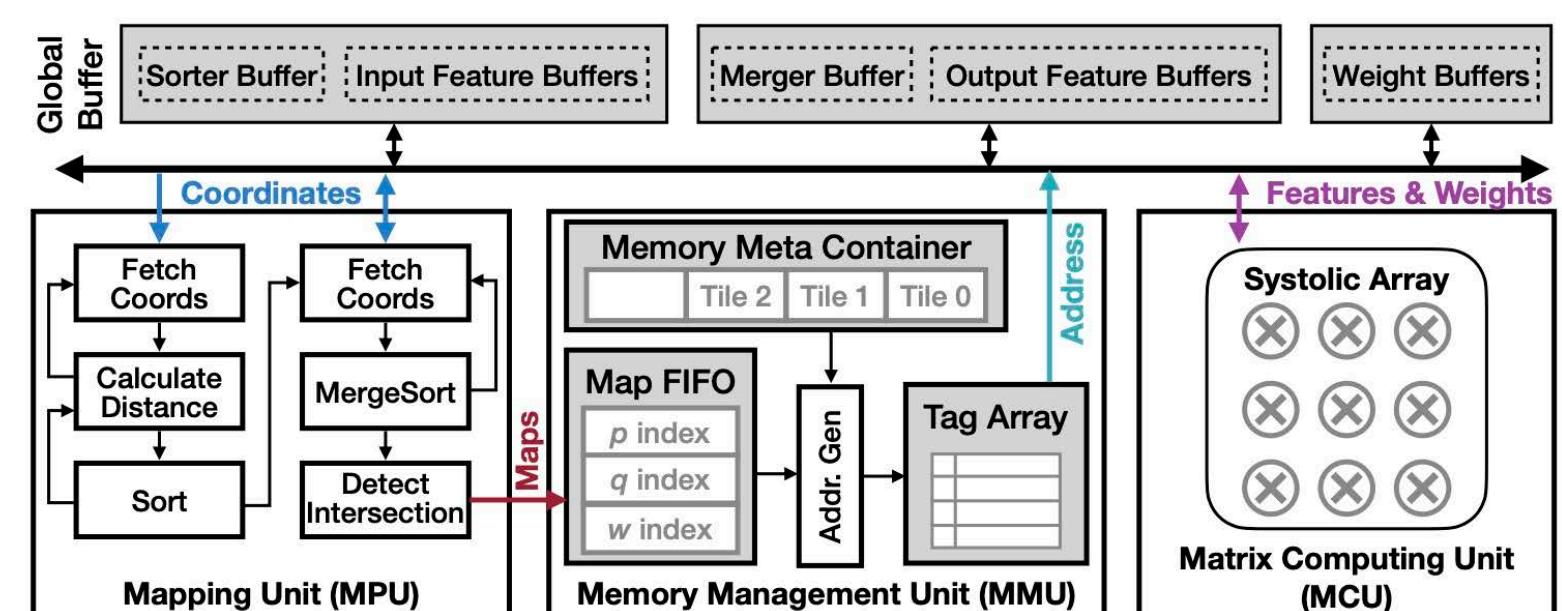
[ESE](#), Han et al, FPGA'17



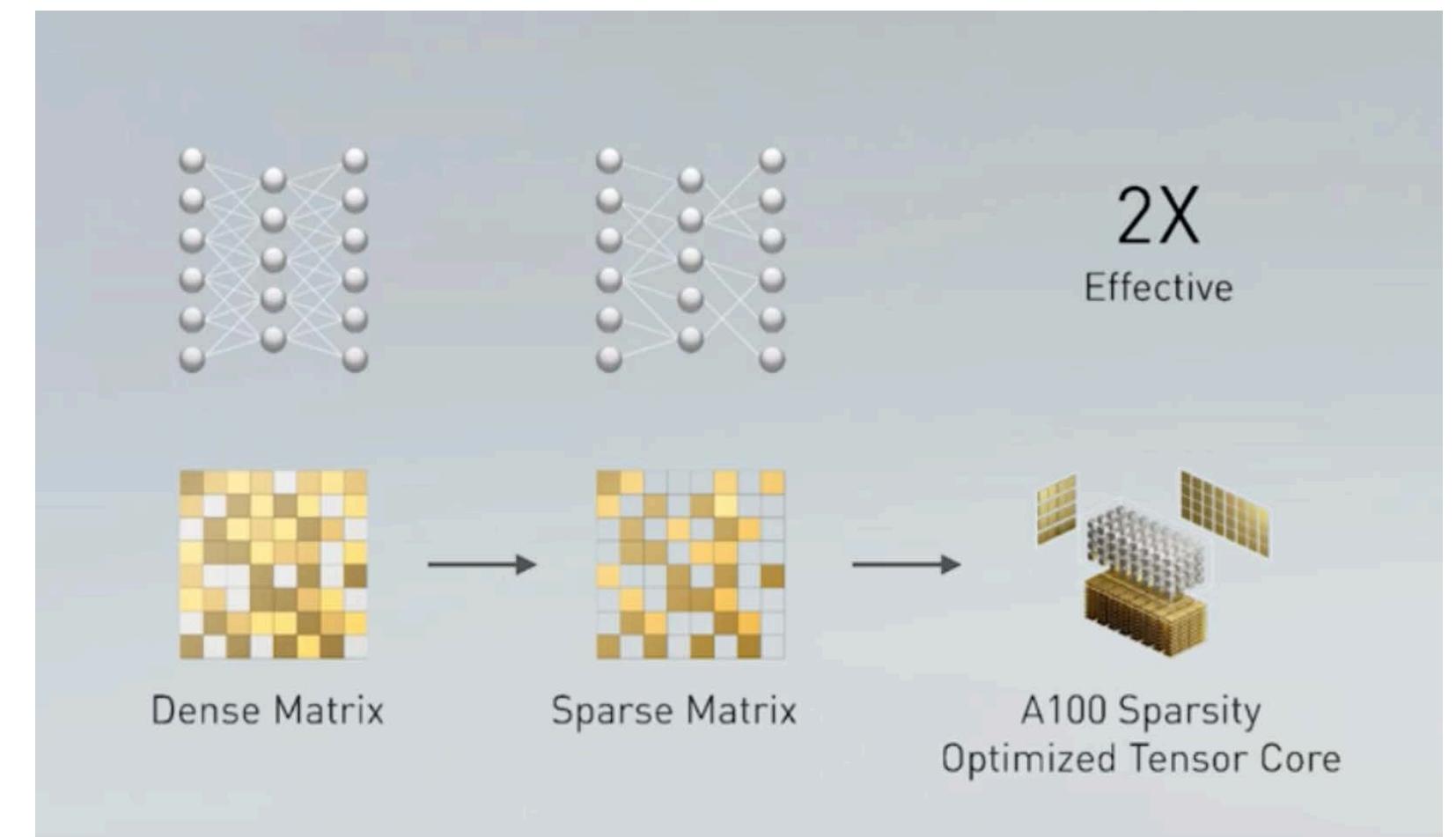
[SpArch](#), Wang et al, HPCA'20



[SpAtten](#), Wang et al, HPCA'21

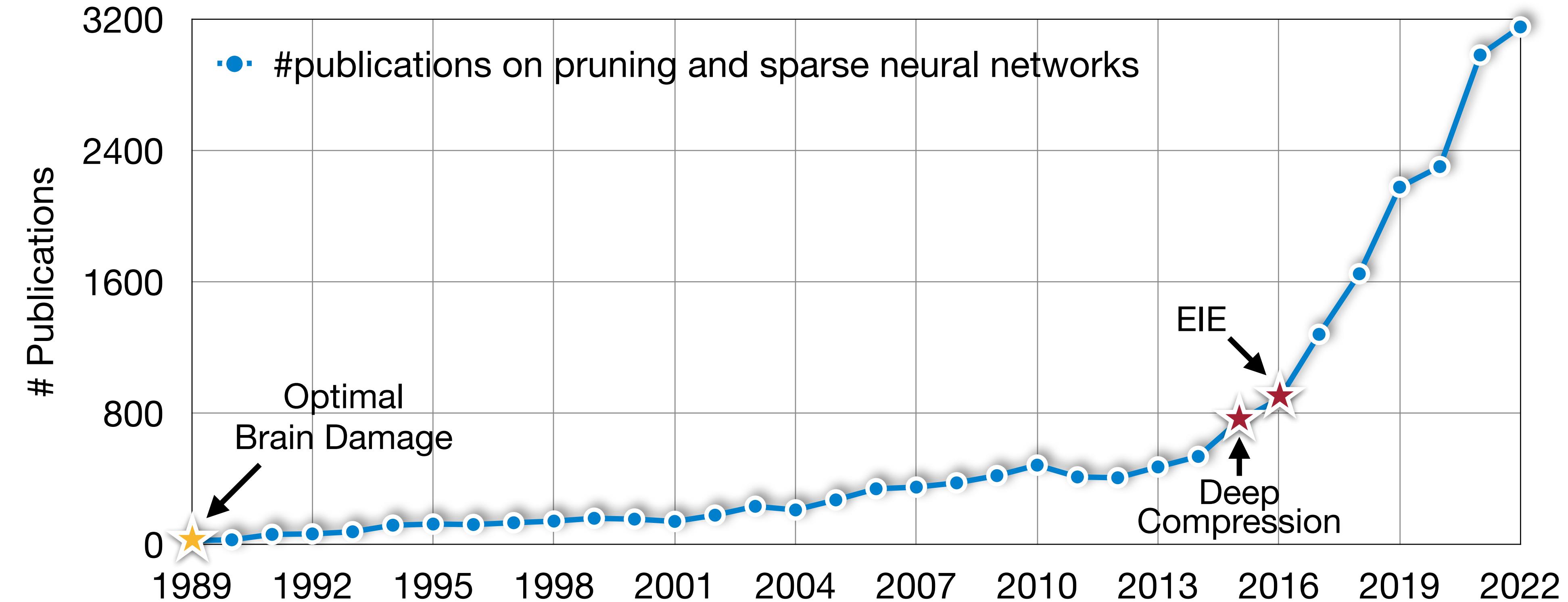


[PointAcc](#), Lin et al, Micro'21



NVIDIA Ampere Architecture

# #publications in pruning and sparsity



The number of publications on neural network pruning and sparsity quickly increased since 2015, including both algorithms and systems.

Source: <https://github.com/mit-han-lab/pruning-sparsity-publications>

# Top-5 most cited papers in 50 years of ISCA

Rank	Citations	Year	Title ( <span style="color: yellow;">★</span> means it won the <i>ISCA Influential Paper Award</i> )	First Author + HOF Authors	Type	Topic
1	5351	1995	<a href="#">The SPLASH-2 programs: Characterization and methodological considerations</a>	<a href="#">Stephen Woo</a> , <a href="#">Anoop Gupta</a>	Tool	Benchmark
2	4214	2017	<a href="#">In-datacenter performance analysis of a Tensor Processing Unit</a>	<a href="#">Norm Jouppi</a> , <a href="#">David Patterson</a>	Arch	Machine Learning
3	3834	2000	<span style="color: yellow;">★</span> <a href="#">Wattpc: A framework for architectural-level power analysis and optimizations</a>	<a href="#">David Brooks</a> , <a href="#">Margaret Martonosi</a>	Tool	Power
4	3386	1993	<span style="color: yellow;">★</span> <a href="#">Transactional memory: Architectural support for lock-free data structures</a>	<a href="#">Maurice Herlihy</a>	Micro	Parallelism
5	2690	2016	<a href="#">EIE: Efficient inference engine on compressed deep neural network</a>	<a href="#">Song Han</a> , <a href="#">Bill Dally</a> , <a href="#">Mark Horowitz</a>	Arch	Machine Learning

# Retrospective: EIE: Efficient Inference Engine on Sparse and Compressed Neural Network

Song Han<sup>1,3</sup>, Xingyu Liu<sup>4</sup>, Huizi Mao<sup>3</sup>, Jing Pu<sup>5</sup>, Ardavan Pedram<sup>2,6</sup>, Mark A. Horowitz<sup>2</sup>, William J. Dally<sup>2,3</sup>,  
<sup>1</sup>MIT   <sup>2</sup>Stanford   <sup>3</sup> NVIDIA   <sup>4</sup> CMU   <sup>5</sup> Google   <sup>6</sup> Samsung

The first principle of efficient AI computing is to be lazy: avoid redundant computation, quickly reject the work, or delay the work.

- Generative AI: spatial sparsity [SIGE, NeurIPS'22]
- Transformer: token sparsity, progressive quantization [SpAtten, HPCA'21]
- Video: temporal sparsity [TSM, ICCV'19]
- Point cloud: spatial sparsity [TorchSparse, MLSys'22 & PointAcc, Micro'22]

We envision future AI models will be sparse at various granularity and structures. Co-designed with specialized accelerators, sparse models will become more efficient and accessible.

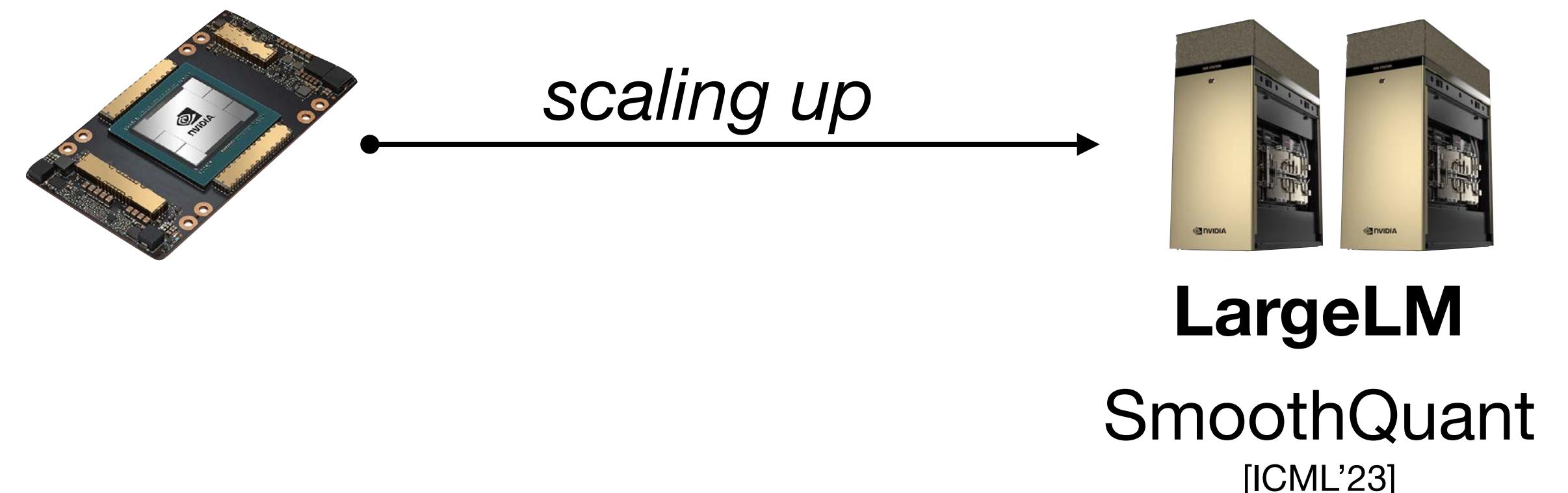
# Overview

## Efficient deep learning computing



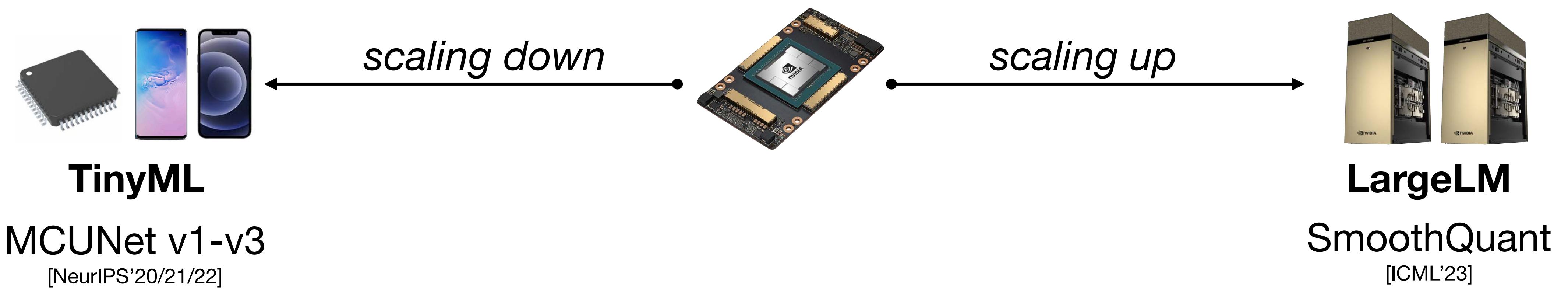
# Overview

## Efficient deep learning computing



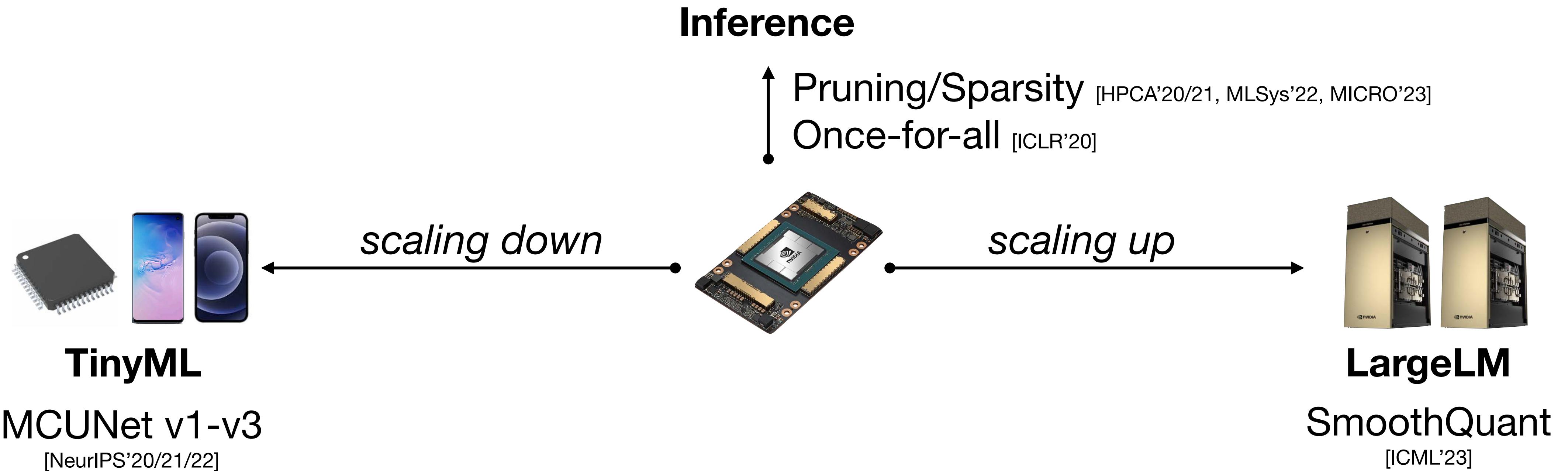
# Overview

## Efficient deep learning computing



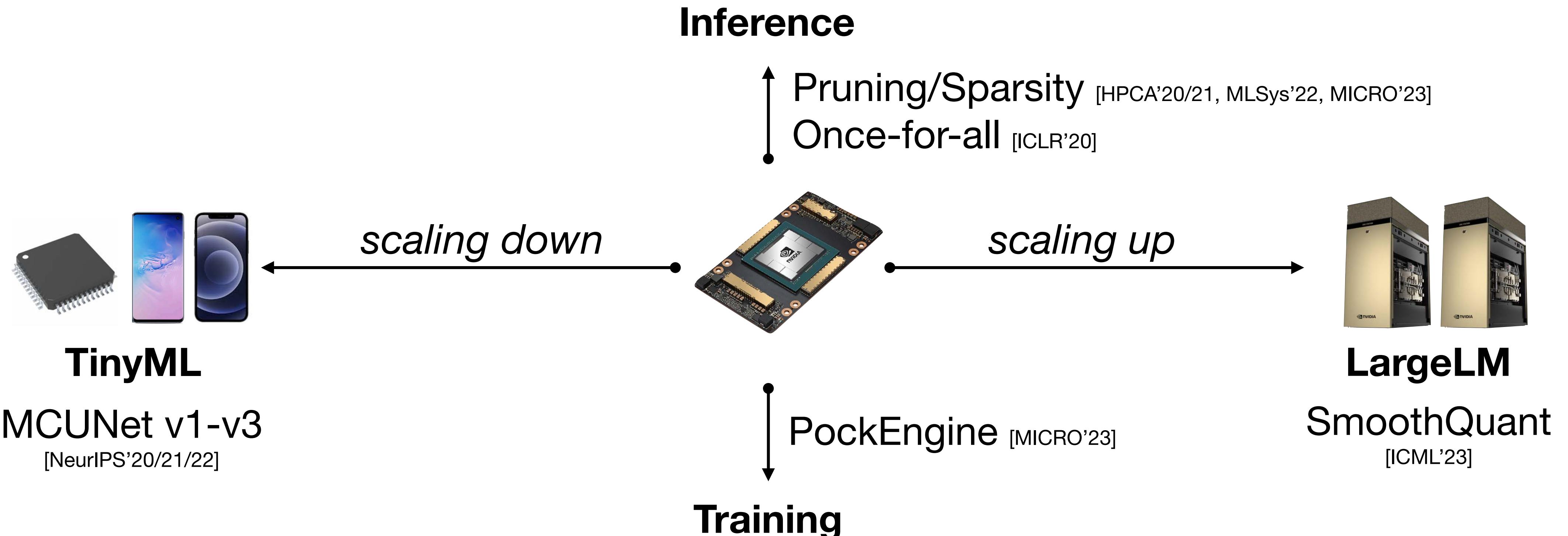
# Overview

## Efficient deep learning computing



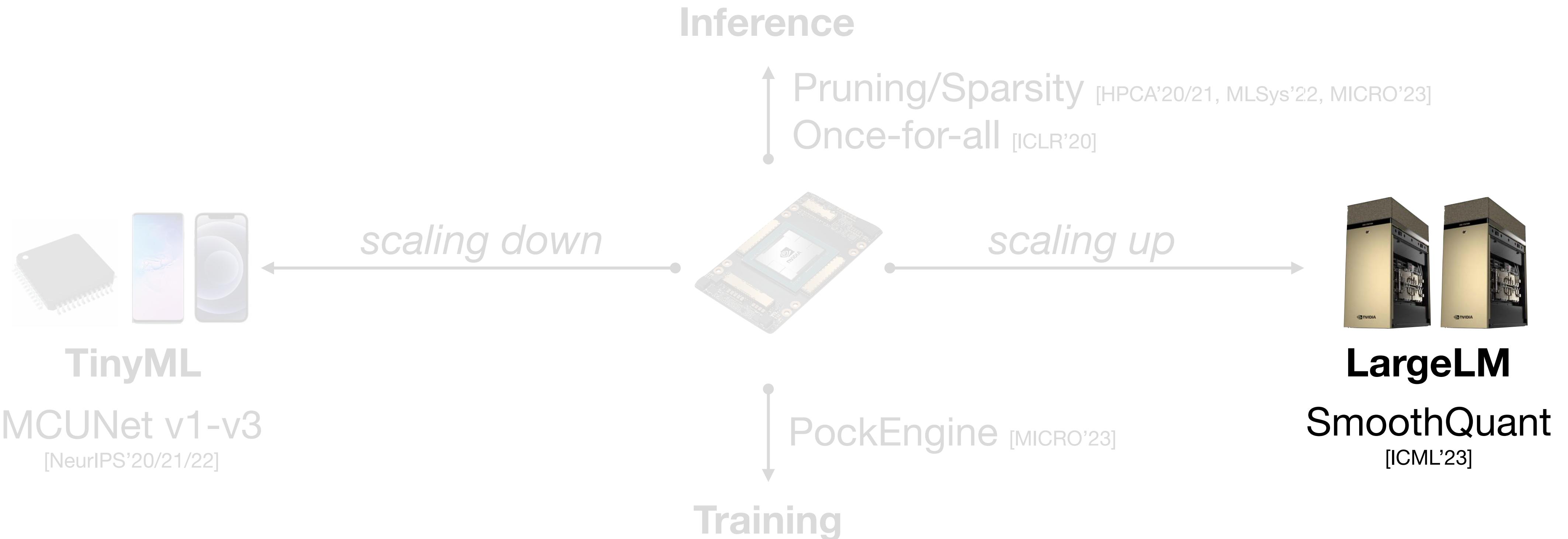
# Overview

## Efficient deep learning computing



# Overview

## Efficient deep learning computing



# Efficient Large Language Models

LLM serving cost is extremely high

We're experiencing exceptionally high demand. Please hang tight as we work on scaling our systems. 

**ChatGPT is at capacity right now**

[Get notified when we're back](#)

**Write a rap about the status of ChatGPT.**

Yeah, yo, what's up ChatGPT fam  
A lot of people trying to jam  
But don't worry, we got your back  
Just check back soon, we'll get on track

ChatGPT's the place to be  
For all your AI chat needs  
We're working hard to keep up the pace  
So hold tight, we'll be back in this space

So don't you worry, don't be mad  
We're working hard, it's not so bad  
Just give us time, we'll be back  
ChatGPT, the future of chat, that's a fact.

**Your Account**

**Free Plan**

Your Current Plan

- Available when demand is low
- Standard response speed
- Regular model updates

**ChatGPT Plus USD \$20/mo**

Upgrade plan

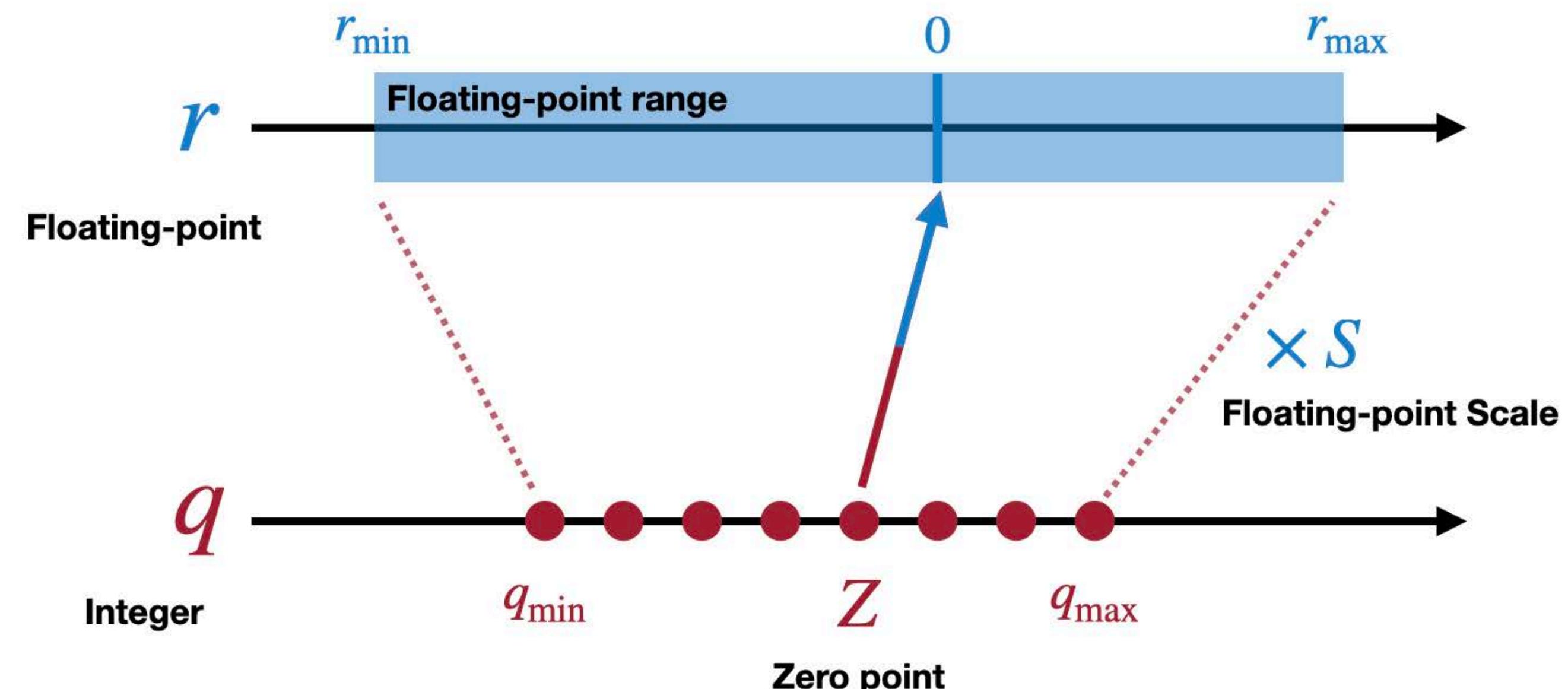
Due to high demand, we've temporarily paused upgrades.

Priority access to new features



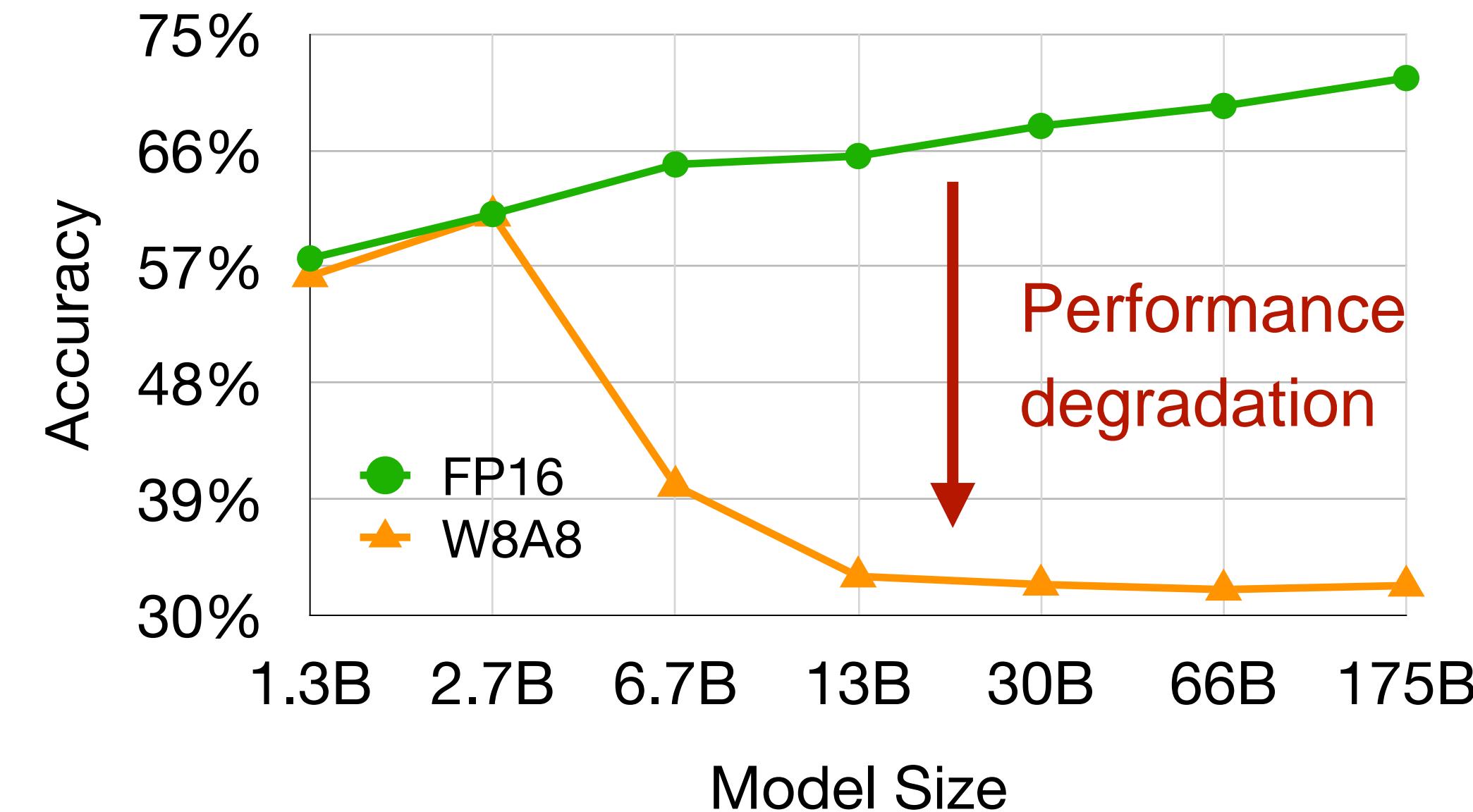
# Quantization Can Reduce Deployment Costs

- Serving a 175B GPT-3 model at least requires:
  - FP16: 350GB memory  $\rightarrow$  5 x 80GB A100 GPUs
  - INT8: 175GB memory  $\rightarrow$  3 x 80GB A100 GPUs



# SmoothQuant

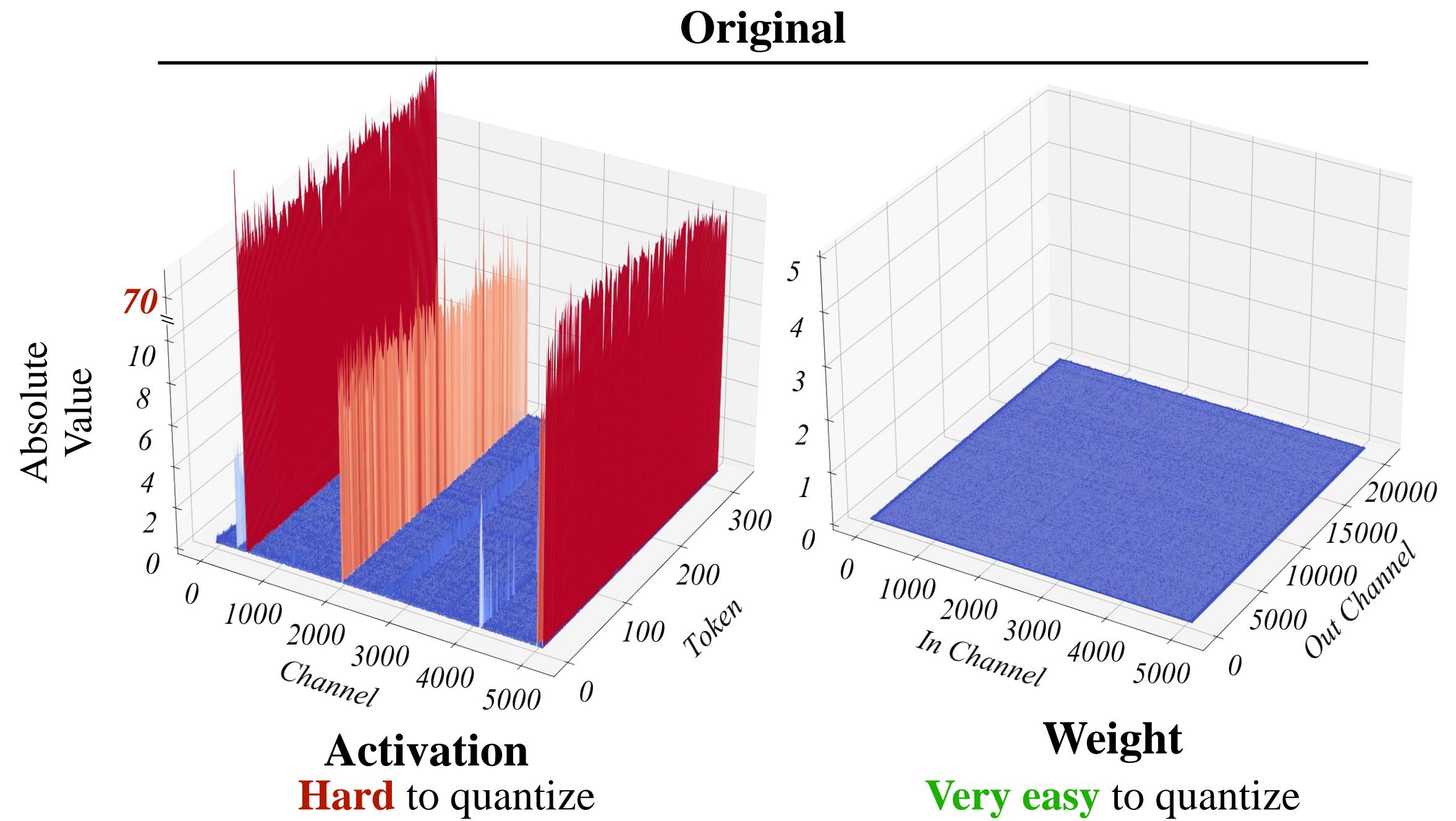
## Traditional Quantization Methods Degrades the Accuracy of LLM



- INT8 quantization has been an industry standard for CNNs, but not LLM.
- When model size > 7 Billion parameters, systematic outliers emerge.
- Traditional quantization methods destroy the accuracy.

# SmoothQuant

## Smoothing activation to reduce quantization error

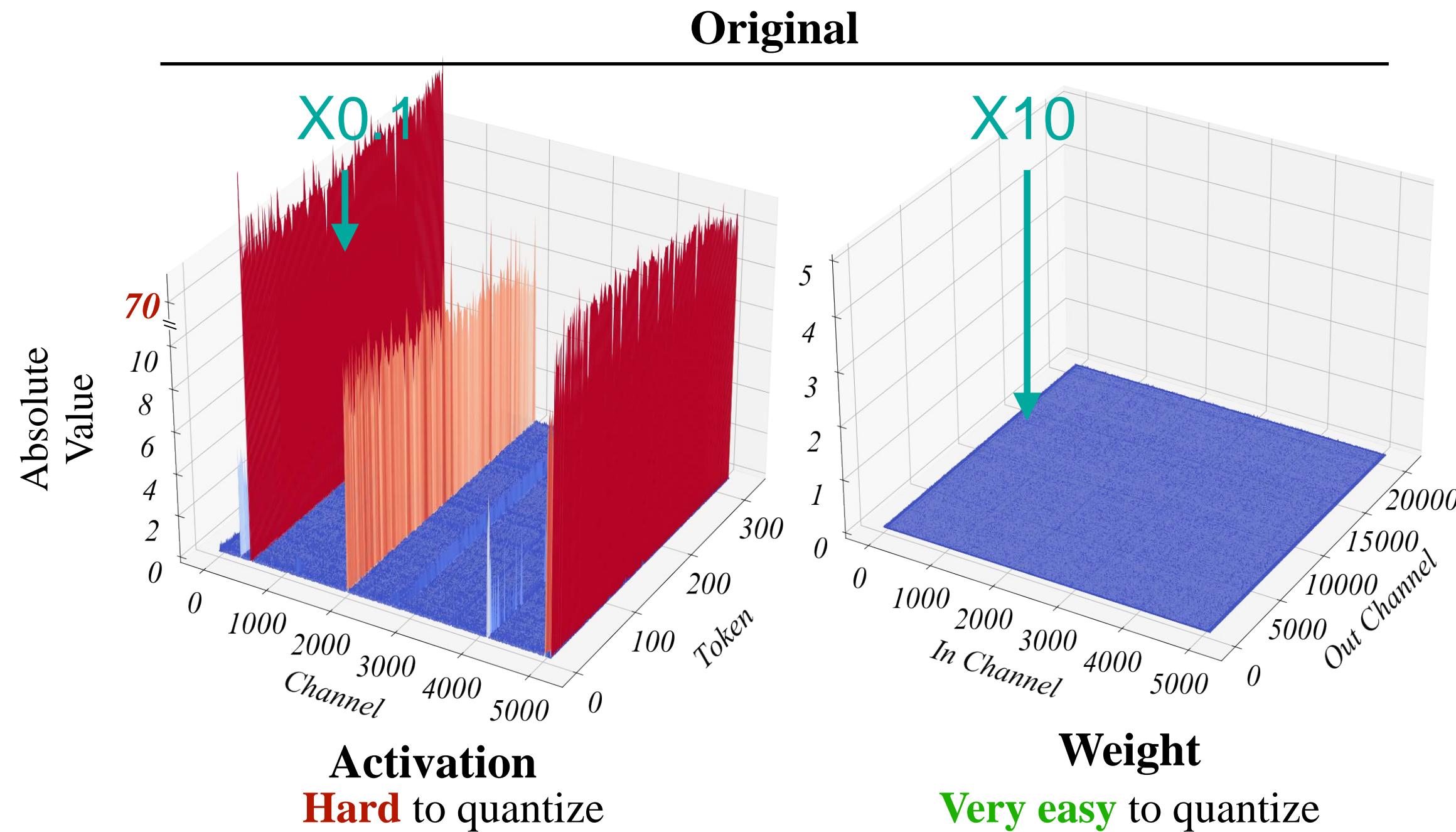


**Weight**  
**Very easy** to quantize

- Weights are easy to quantize, but activation is hard due to outliers
- Luckily, outliers persist in fixed channels

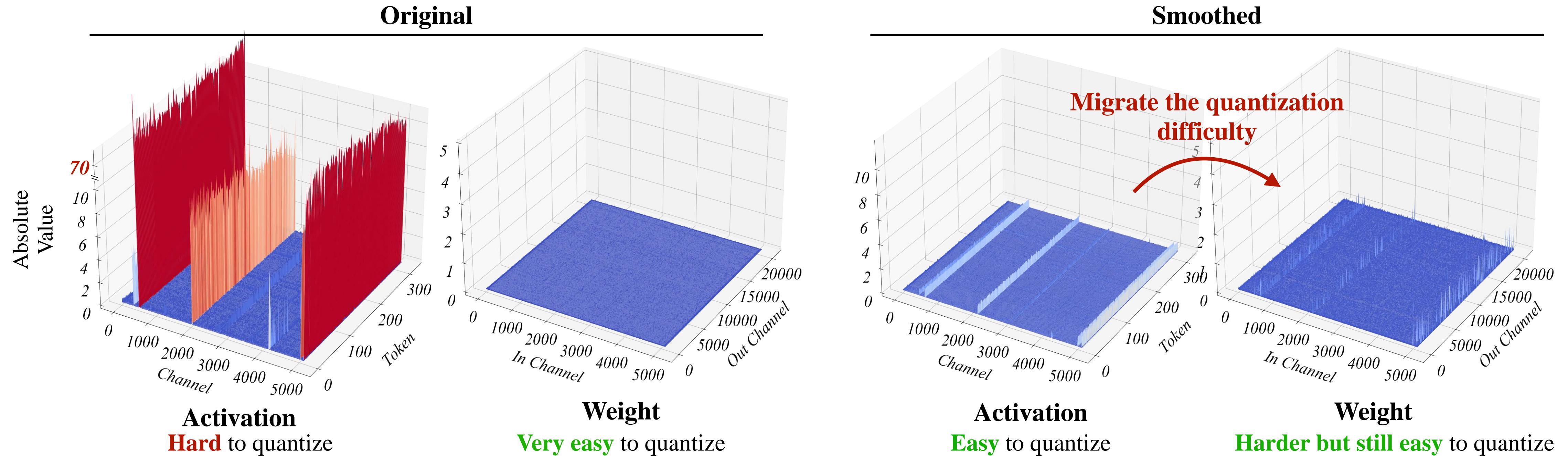
# SmoothQuant

## Smoothing activation to reduce quantization error



# SmoothQuant

## Smoothing activation to reduce quantization error



- Weights are easy to quantize, but activation is hard due to outliers
- Luckily, outliers persist in fixed channels
- Migrate the quantization difficulty from activation to weights, so both are easy to quantize

# SmoothQuant

Smoothing activation to reduce quantization error

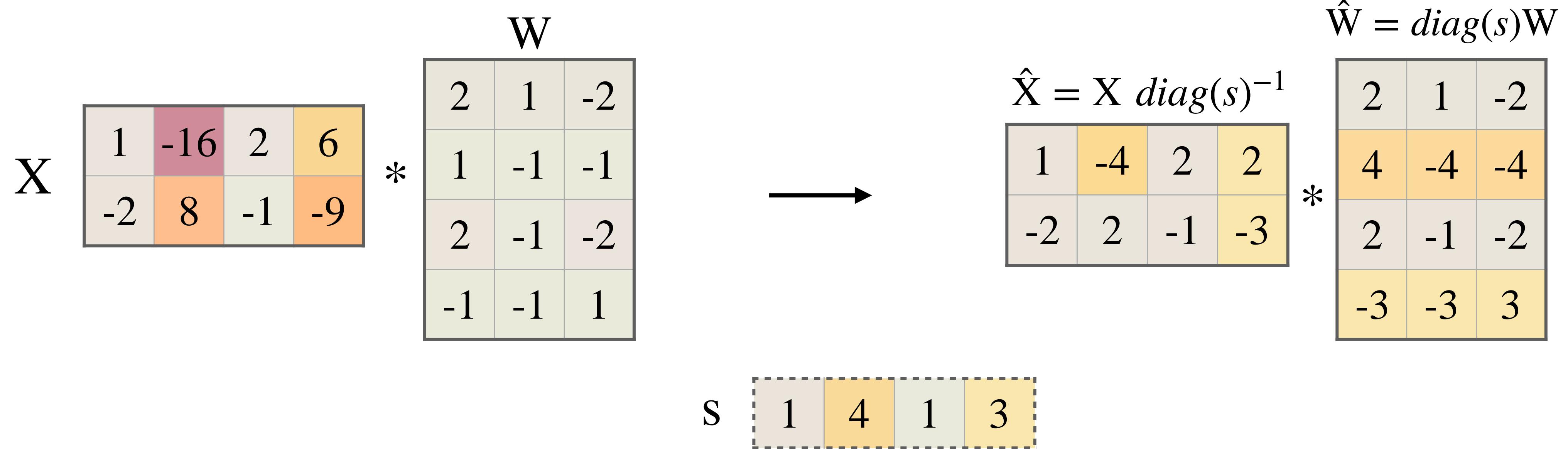
$$\mathbf{X} \quad \begin{matrix} 1 & -16 & 2 & 6 \\ -2 & 8 & -1 & -9 \end{matrix} \quad * \quad \begin{matrix} \mathbf{W} \\ \begin{matrix} 2 & 1 & -2 \\ 1 & -1 & -1 \\ 2 & -1 & -2 \\ -1 & -1 & 1 \end{matrix} \end{matrix} \quad \begin{matrix} \mathbf{S} \\ \begin{matrix} 1 & 4 & 1 & 3 \end{matrix} \end{matrix}$$

$$\mathbf{s}_j = \max(\mathbf{X}_j)^\alpha / \max(\mathbf{W}_j)^{1-\alpha}, j = 1, 2, \dots, C_i$$

$$\mathbf{Y} = (\mathbf{X} \text{diag}(\mathbf{s})^{-1}) \cdot (\text{diag}(\mathbf{s}) \mathbf{W}) = \hat{\mathbf{X}} \hat{\mathbf{W}}$$

# SmoothQuant

Smoothing activation to reduce quantization error



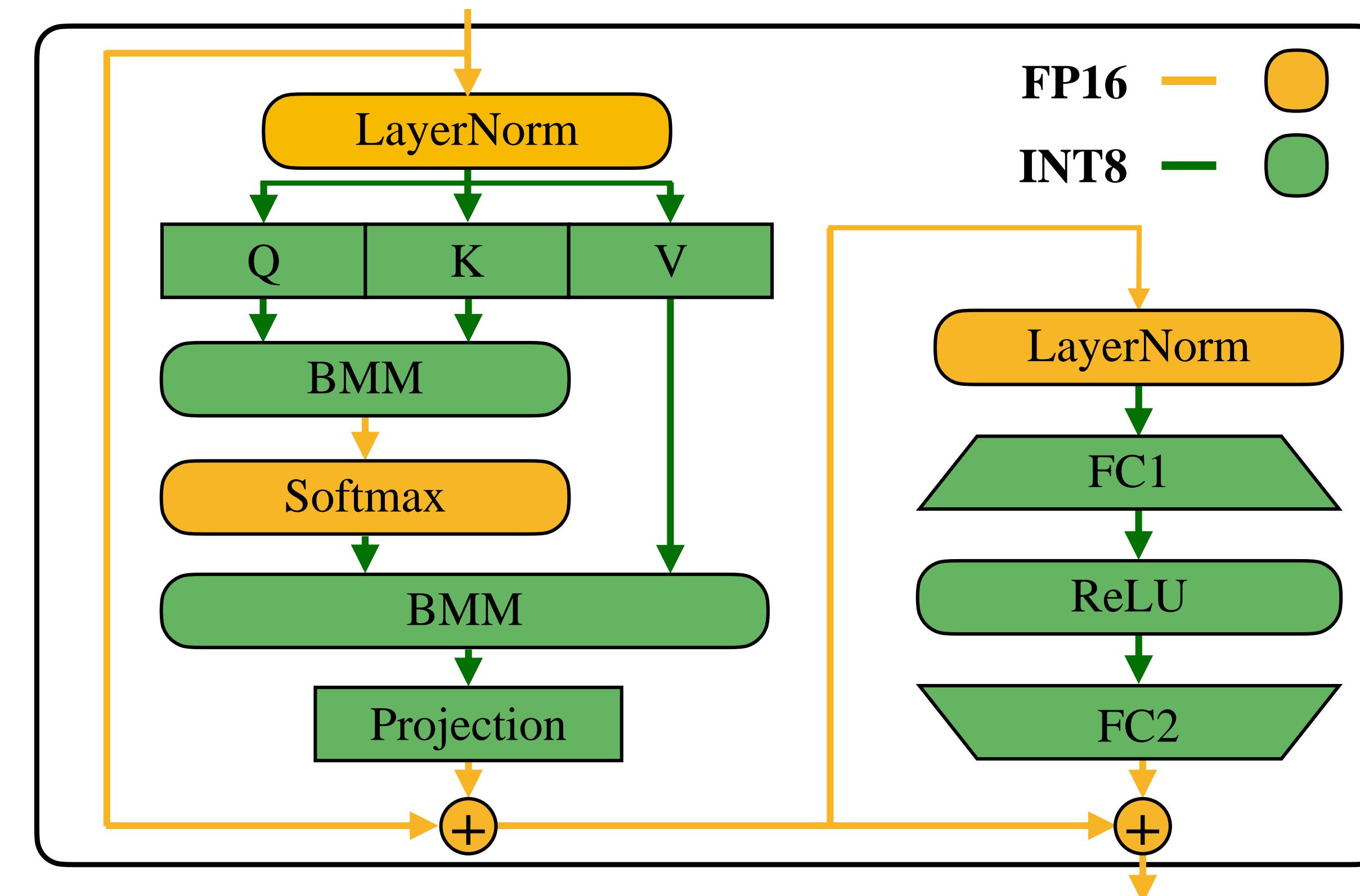
$$s_j = \max(|X_j|)^\alpha / \max(|W_j|)^{1-\alpha}, j = 1, 2, \dots, C_i$$

$$Y = (X \ diag(s)^{-1}) \cdot (diag(s)W) = \hat{X} \hat{W}$$

# SmoothQuant

## Efficient System Implementation

- All compute-intensive operators (Linears, BMMs) are quantized



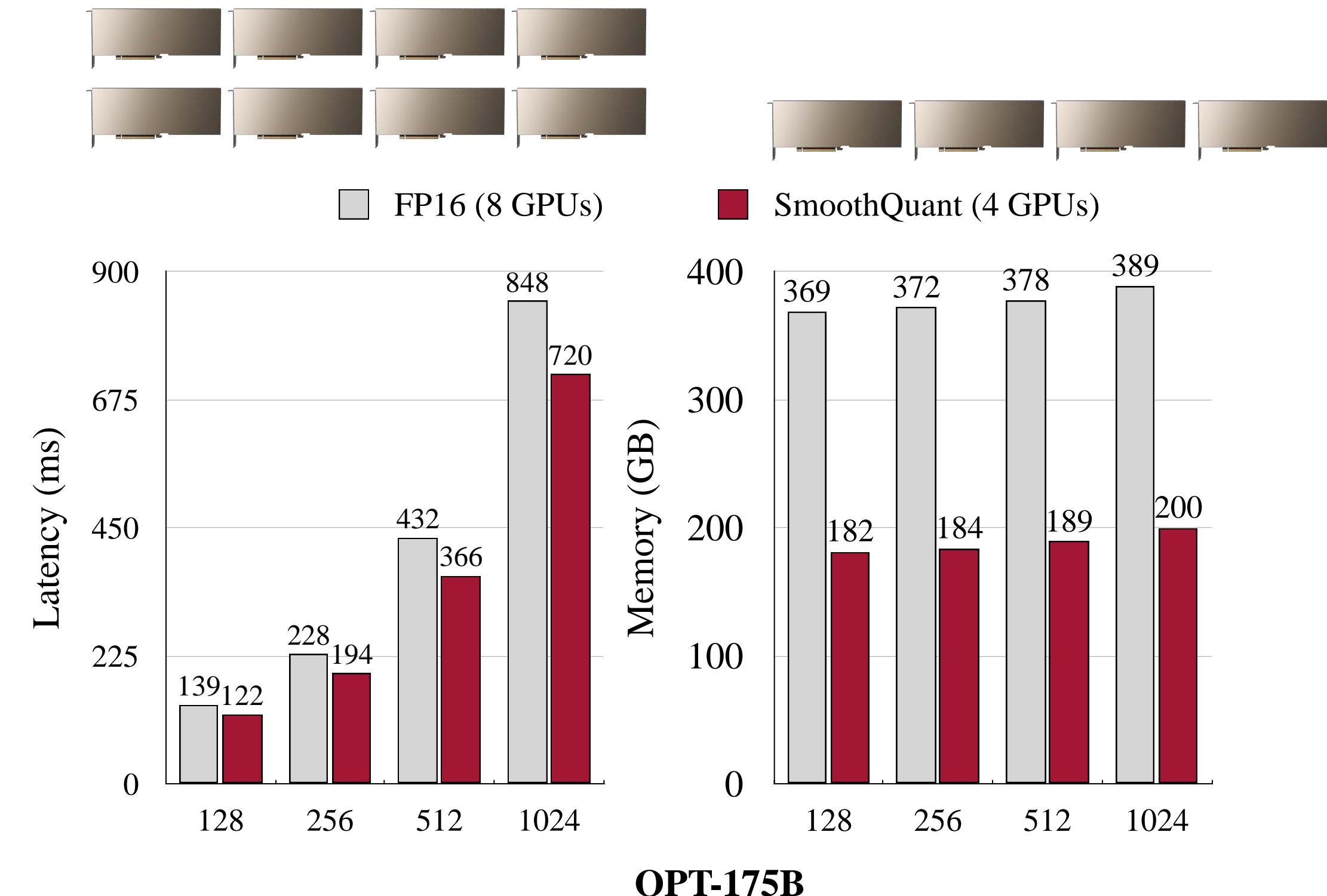
# SmoothQuant (W8A8)

## Accurate and efficient quantization of various LLMs

- SmoothQuant well maintains the accuracy without fine-tuning.
- SmoothQuant can both accelerate inference and halve the memory footprint.

LAMBADA Accuracy

	<b>OPT-175B</b>	<b>BLOOM-176B</b>	<b>GLM-130B</b>
FP16	71.6%	68.2%	73.8%
SmoothQuant	71.2%	68.3%	73.7%



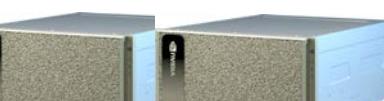
# SmoothQuant (W8A8)

## Scaling up: W8A8 quantization of MT-NLG 530B

MT-NLG 530B Accuracy

	LAMBADA	HellaSwag	PIQA	WinoGrande	Average
FP16	76.6%	62.1%	81.0%	72.9%	73.1%
INT8	77.2%	60.4%	80.7%	74.1%	73.1%

MT-NLG 530B Efficiency

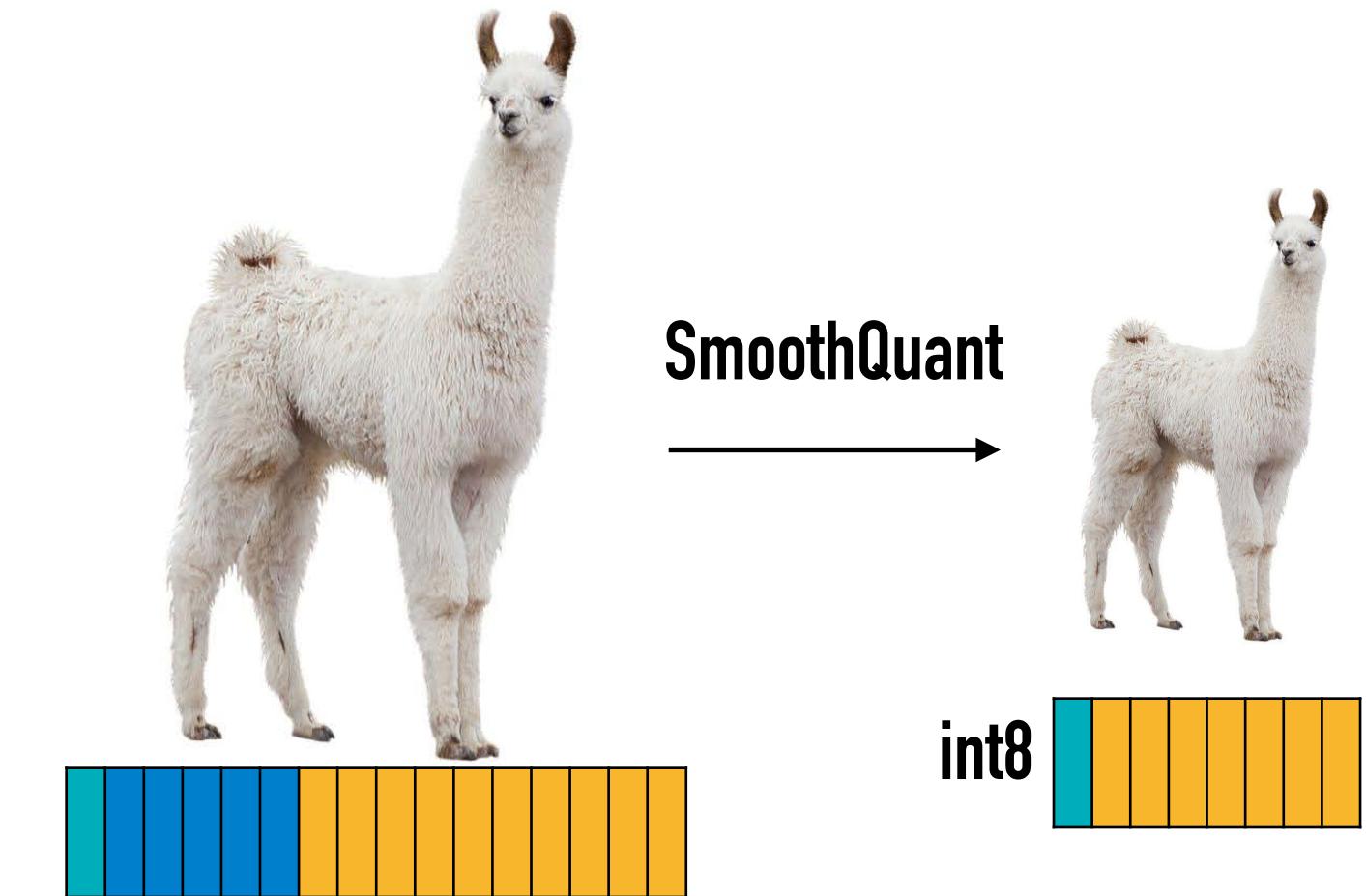
	SeqLen	Prec.	#GPUs	Latency	Memory	
512	FP16	16	16	838ms	1068GB	
		8	8	839ms	545GB	
1024	FP16	16	16	1707ms	1095GB	
		8	8	1689ms	570GB	

- SmoothQuant can accurately quantize MT-NLG 530B model and reduce the serving GPU numbers by half at a similar latency, which allows serving the 530B model within a single node.

# SmoothQuant (W8A8)

## Advancing new efficient open model LLaMA

- **LLaMA** (and its successors like Alpaca) are popular open-source LLMs, which introduced SwishGLU, making activation quantization even harder
- SmoothQuant can losslessly quantize LLaMA families, further lowering the hardware barrier



Wikitext PPL↓	LLaMA 7B	LLaMA 13B	LLaMA 30B	LLaMA 65B
FP16	11.51	10.05	7.53	6.17
SmoothQuant	11.56	10.08	7.56	6.20

# Industry & Community Impact

SmoothQuant is widely adopted by industry

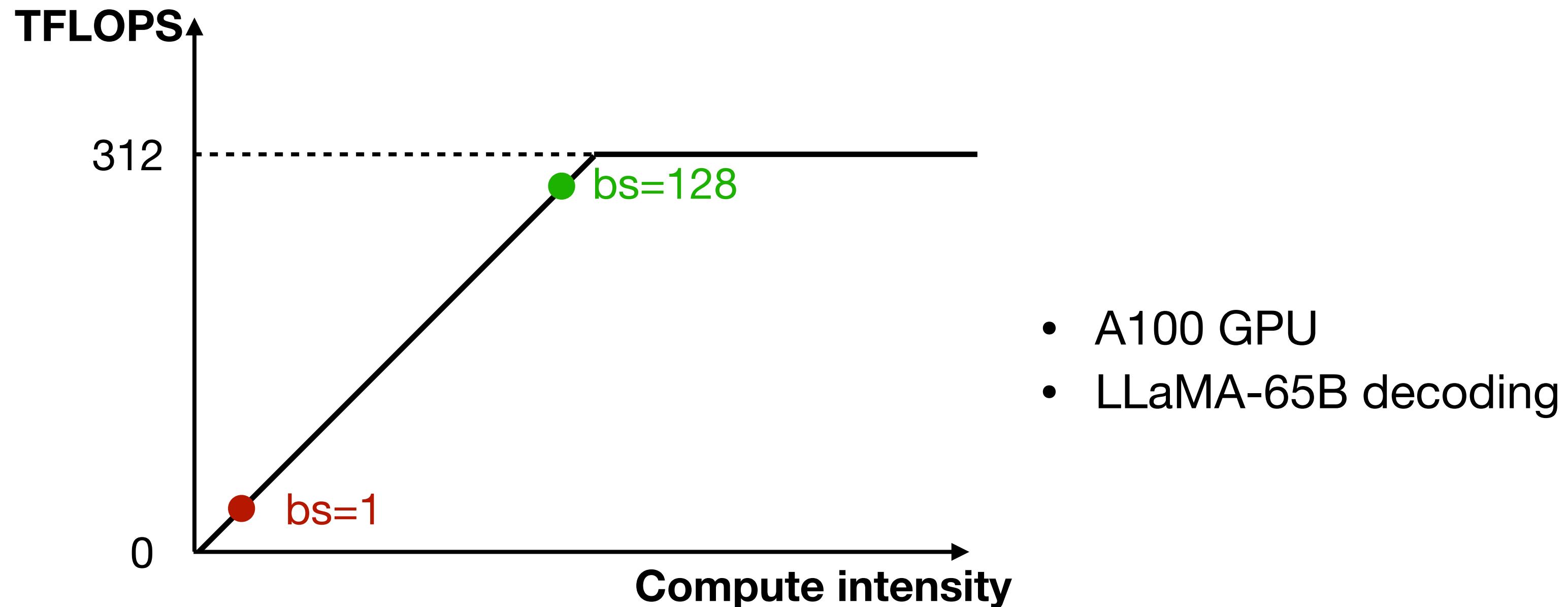
- NVIDIA FasterTransformer
- NVIDIA TRT-LLM
- MLPerf 8-bit: closed the accuracy gap.
- Intel Neural Compressor / Q8-Chat on Xeon
- Ongoing efforts by Meta/Microsoft/Amazon/HuggingFace ...



# W4A16 for Single Query Serving

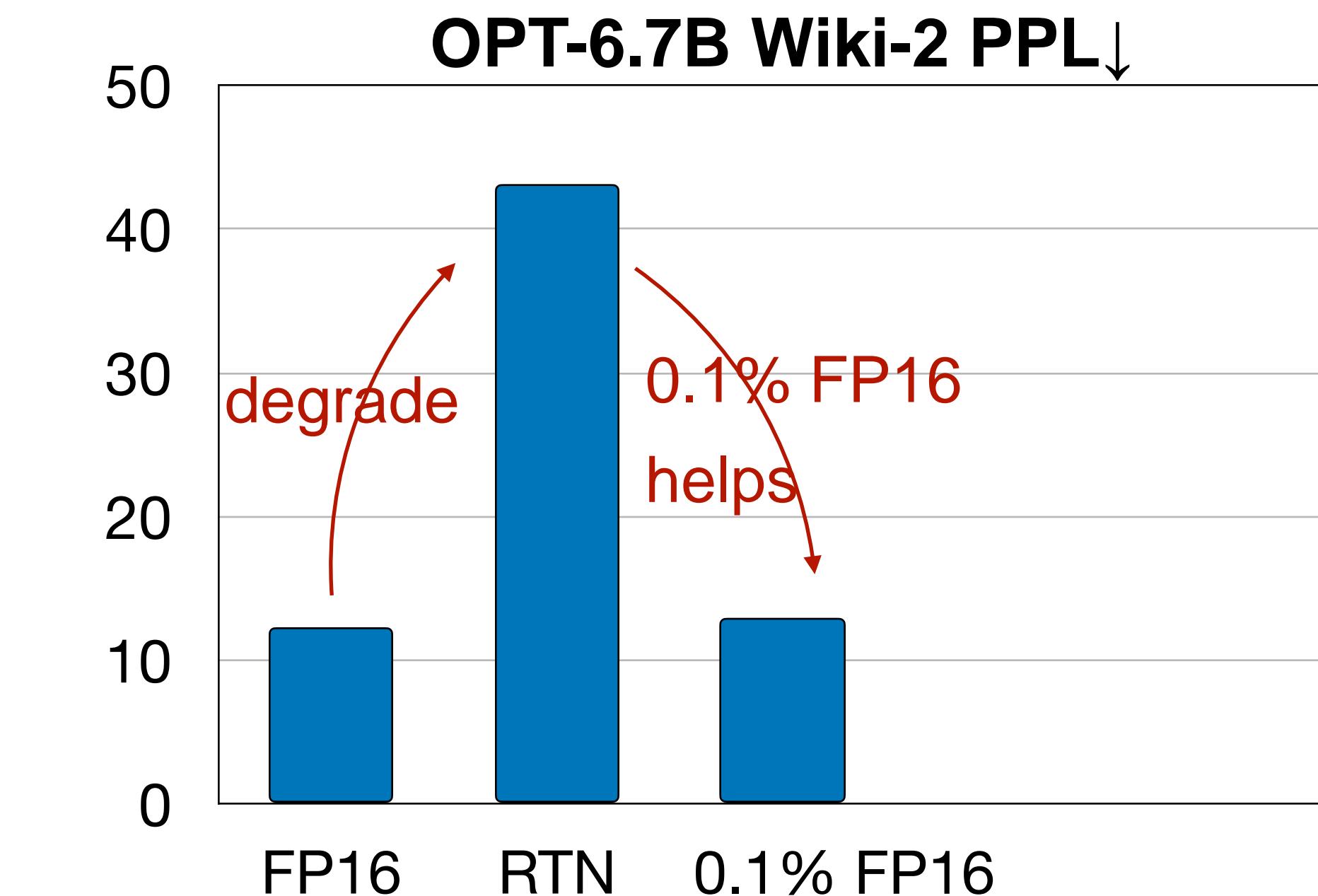
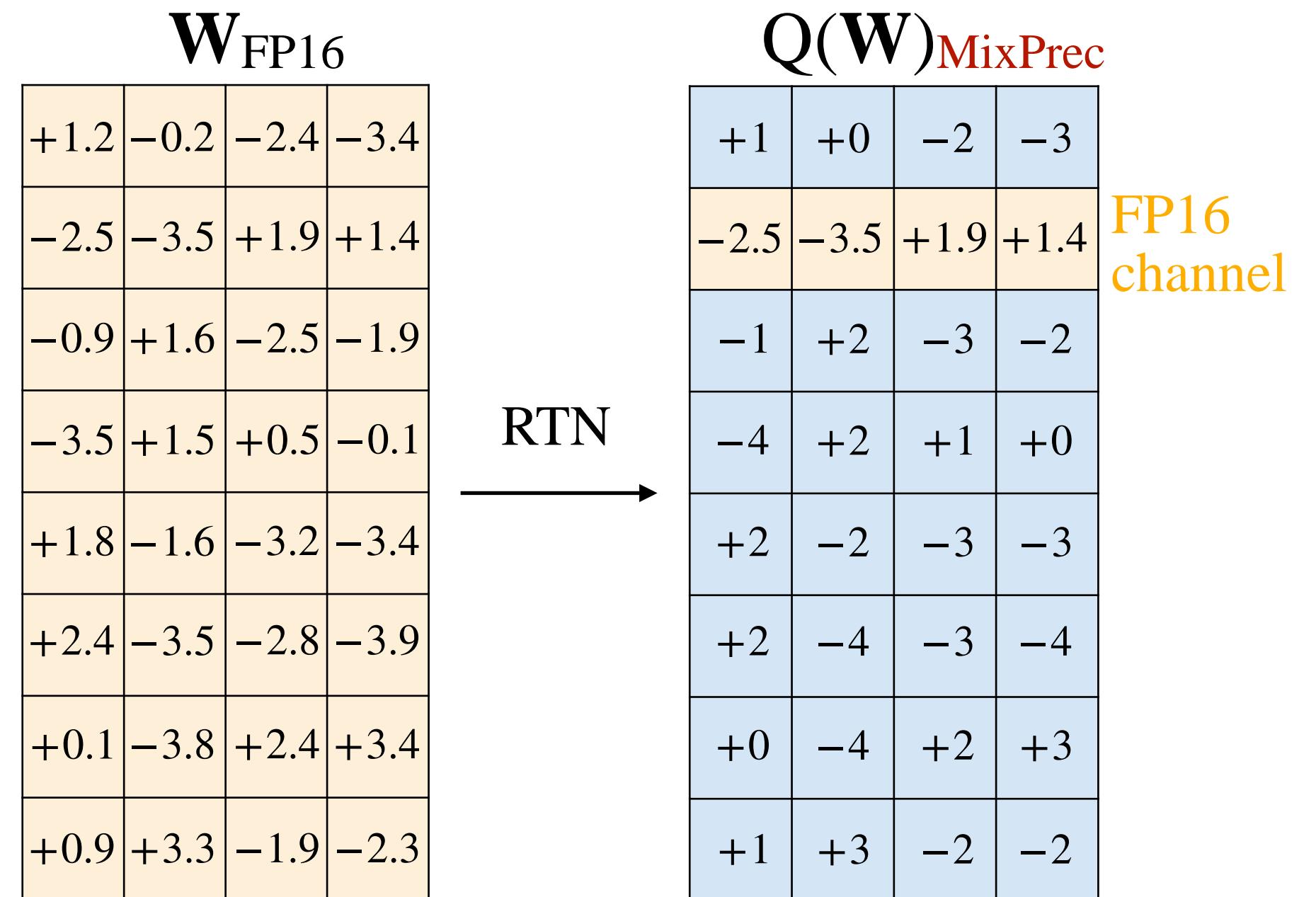
W8A8 cannot address low computational intensity of decoding

- W8A8 quantization is good for batch serving (e.g., batch size 128)
- But single-query LLM inference (e.g., local) is still highly memory-bounded
- We need **low-bit weight-only quantization (e.g., W4A16)** for this setting



# AWQ: Activation-aware Weight Quantization

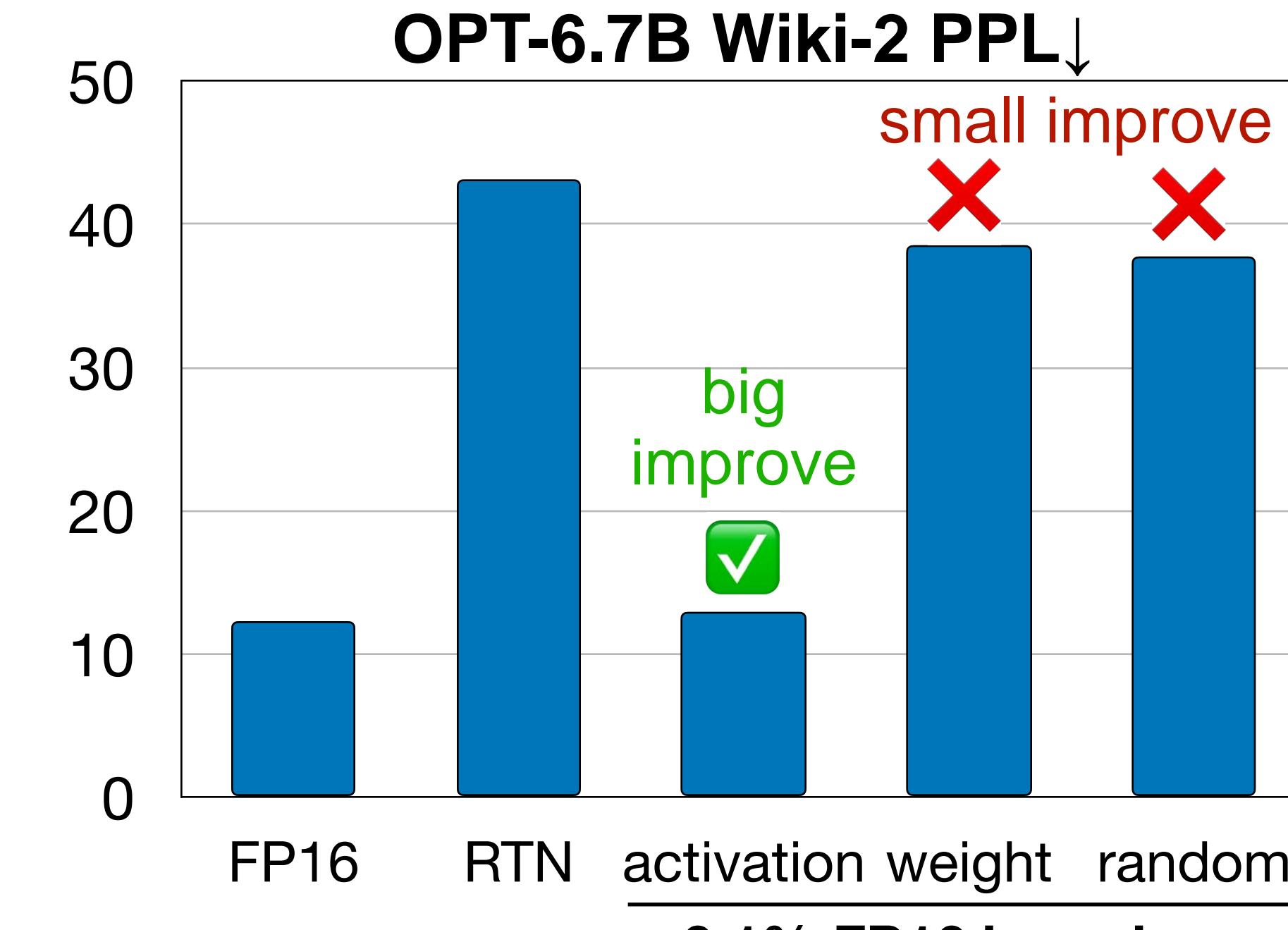
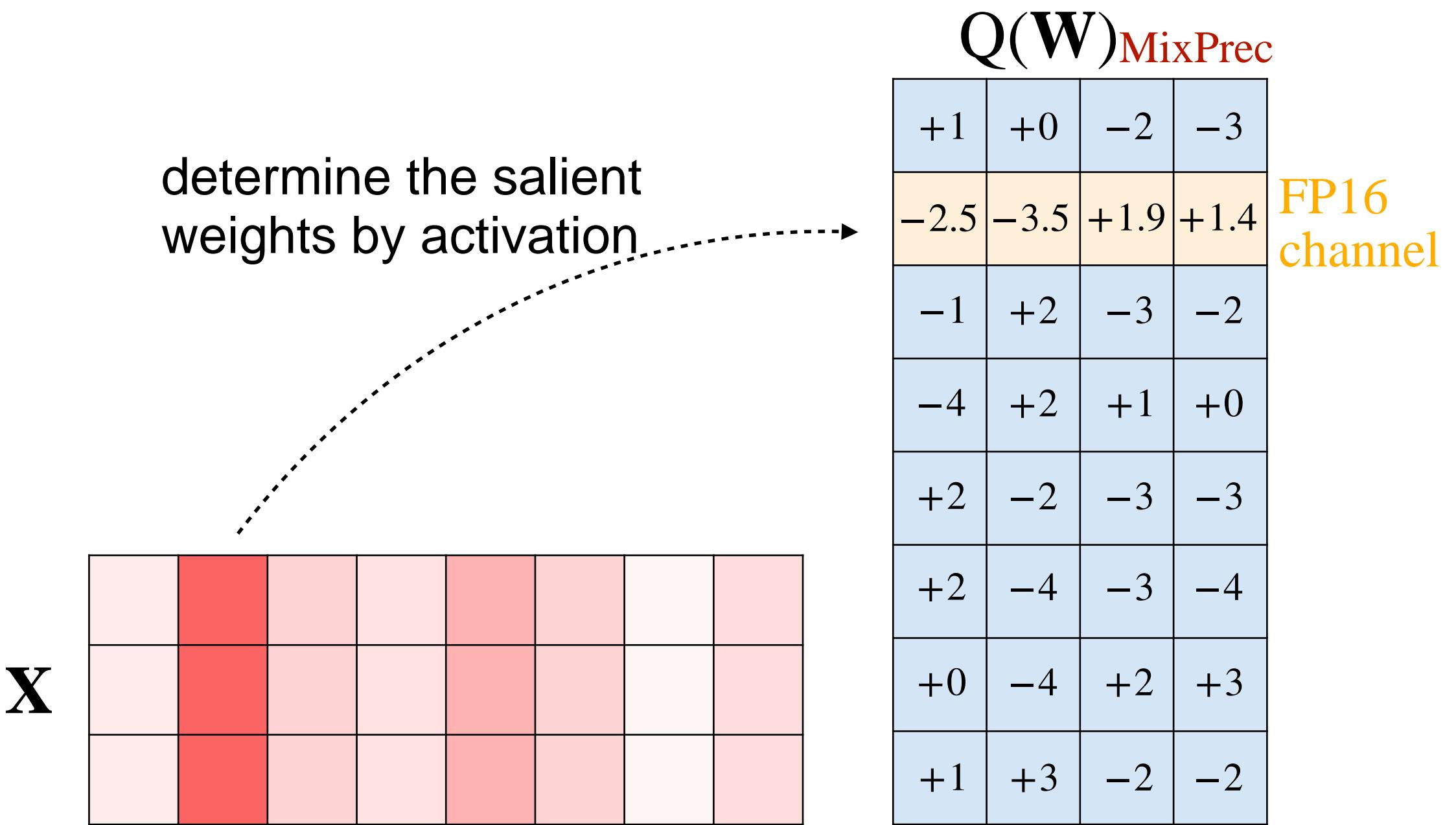
**Observation: Weights are not equally important; 0.1% salient weights**



- We find that weights are not equally important, keeping **only 0.1%** of salient weight channels in FP16 can greatly improve perplexity
- But how do we select salient channels? Should we select based on weight magnitude?

# AWQ: Activation-aware Weight Quantization

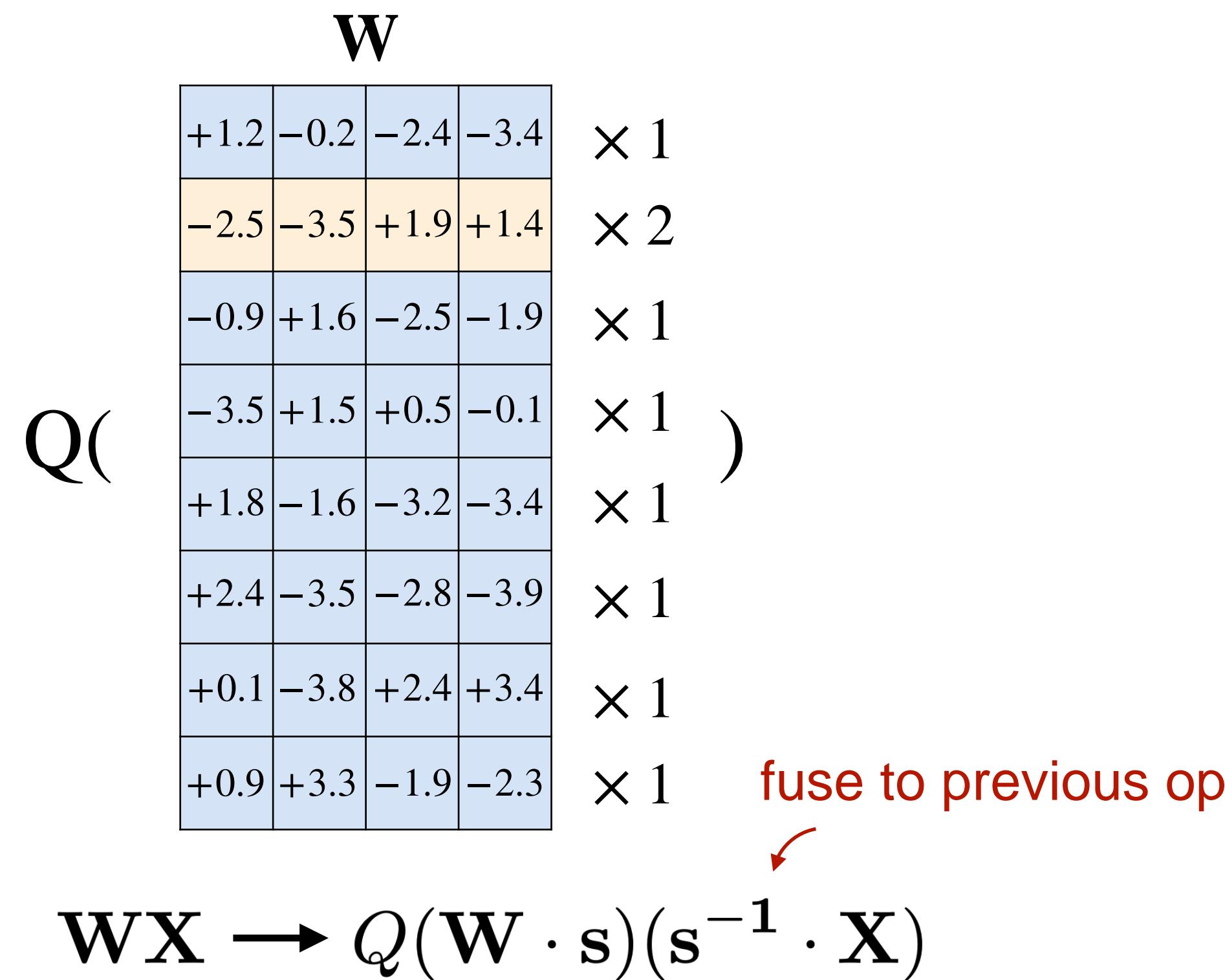
Salient weights are determined by activation distribution, not weight



- We find that weights are not equally important, keeping **only 0.1%** of salient weight channels in FP16 can greatly improve perplexity
- But how do we select salient channels? Should we select based on weight magnitude?
- No! We should look for **activation distribution, but not weight!**

# AWQ: Activation-aware Weight Quantization

Protecting salient weights by scaling (no mixed prec.)



$$Q(\mathbf{w}) \cdot x = \Delta \cdot \text{Round}\left(\frac{\mathbf{w}}{\Delta}\right) \cdot x, \quad \Delta = \frac{\max(\mathbf{w})}{2^{N-1}}$$

$$Q(\mathbf{w} \cdot s)(x/s) = \Delta' \cdot \text{Round}(\mathbf{w}s/\Delta') \cdot x \cdot \frac{1}{s}$$

constant E(error) suppress error

$$\Delta' \approx \Delta, \quad \text{RoundErr} \sim 0.25, \quad s > 1$$

$$\frac{\text{Err}(Q(\mathbf{w} \cdot s)(x/s))}{\text{Err}(Q(\mathbf{w}) \cdot x)} = \frac{\Delta'}{\Delta} \cdot \frac{1}{s} < 1$$

- Multiplying the salient channels with  $s > 1$  reduces its quantization error
- Skip mathematical derivation for now

# Industry & Community Impact

- **SmoothQuant**
  - NVIDIA FasterTransformer
  - NVIDIA TRT-LLM
  - MLPerf 8-bit: closed the accuracy gap.
  - Intel Neural Compressor
  - Ongoing efforts at Meta/Microsoft/Amazon/HuggingFace ...
- **AWQ**
  - AWQ is integrated by [FastChat](#), [vLLM](#), [HuggingFace TGI](#), and [LMDeploy](#).
  - Check out [AutoAWQ](#), a third-party implementation to make AWQ easier to expand to new models, improve inference speed, and integrate into Huggingface



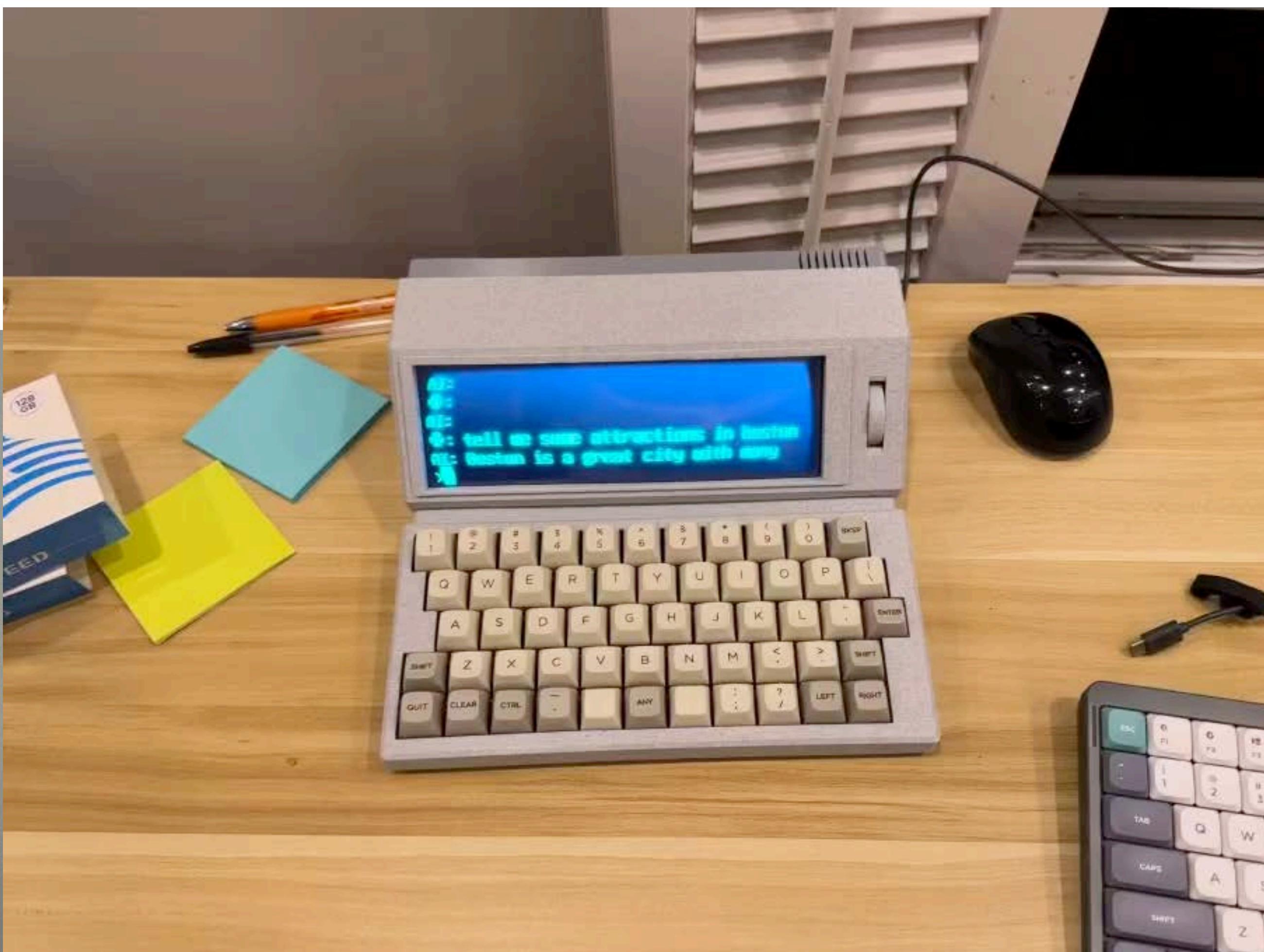
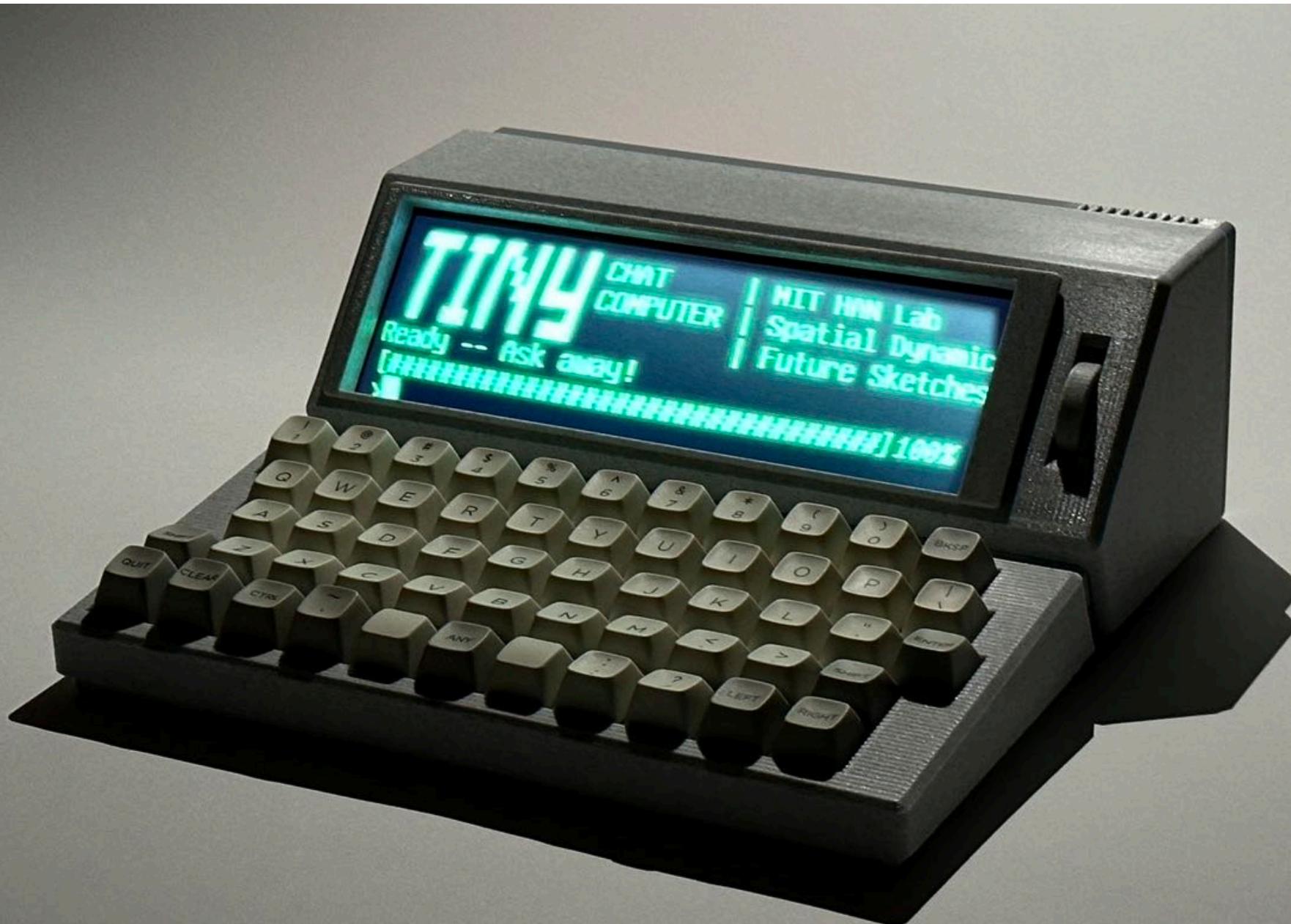
**Hugging Face**



# TinyChat – LLM on the Edge



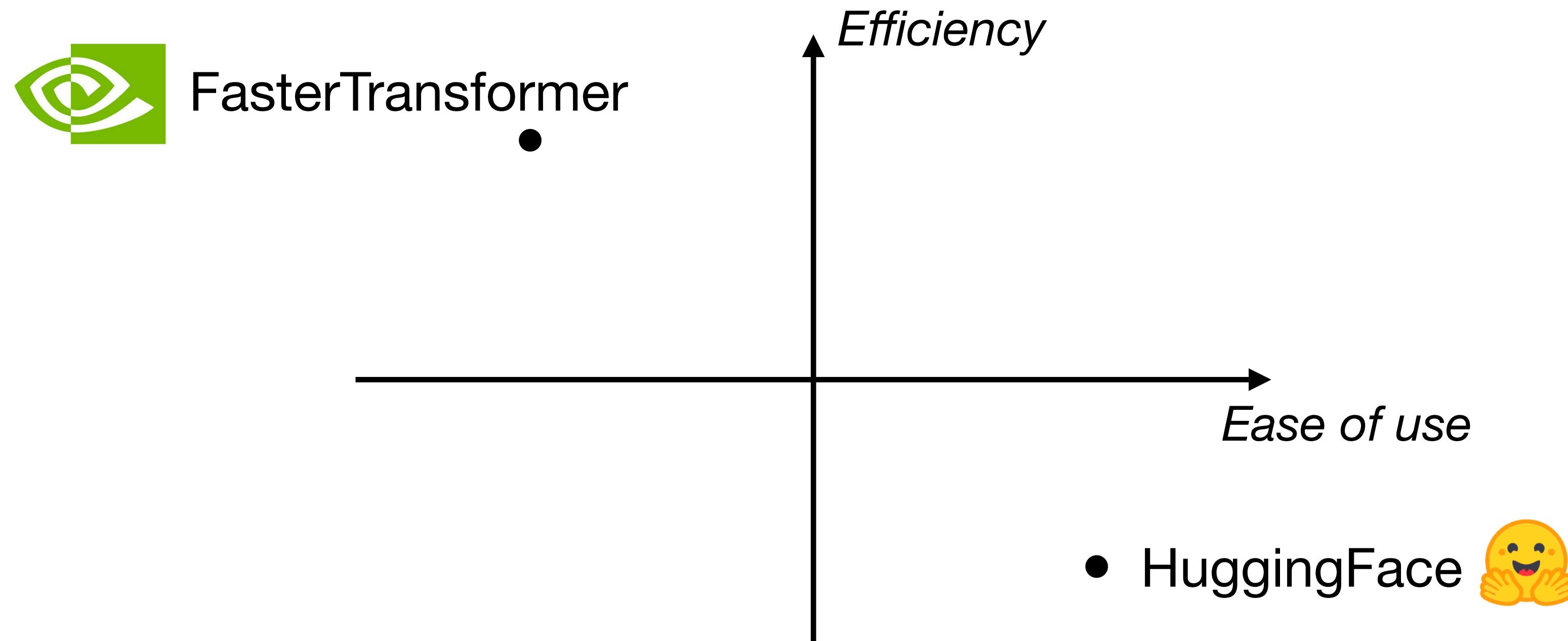
- Deploying LLM on the edge is useful: running **copilot** services (**code** completion, **office**, **game** chat) locally on laptops, cars, robots, and more.
- **resource-constrained**, **low-power** and sometimes **do not have access to the Internet**.
- **Data privacy** is important. Users do not want to share personal data with large companies.



# TinyChat: A Lightweight Serving Infra

## Pythonic, lightweight, efficient

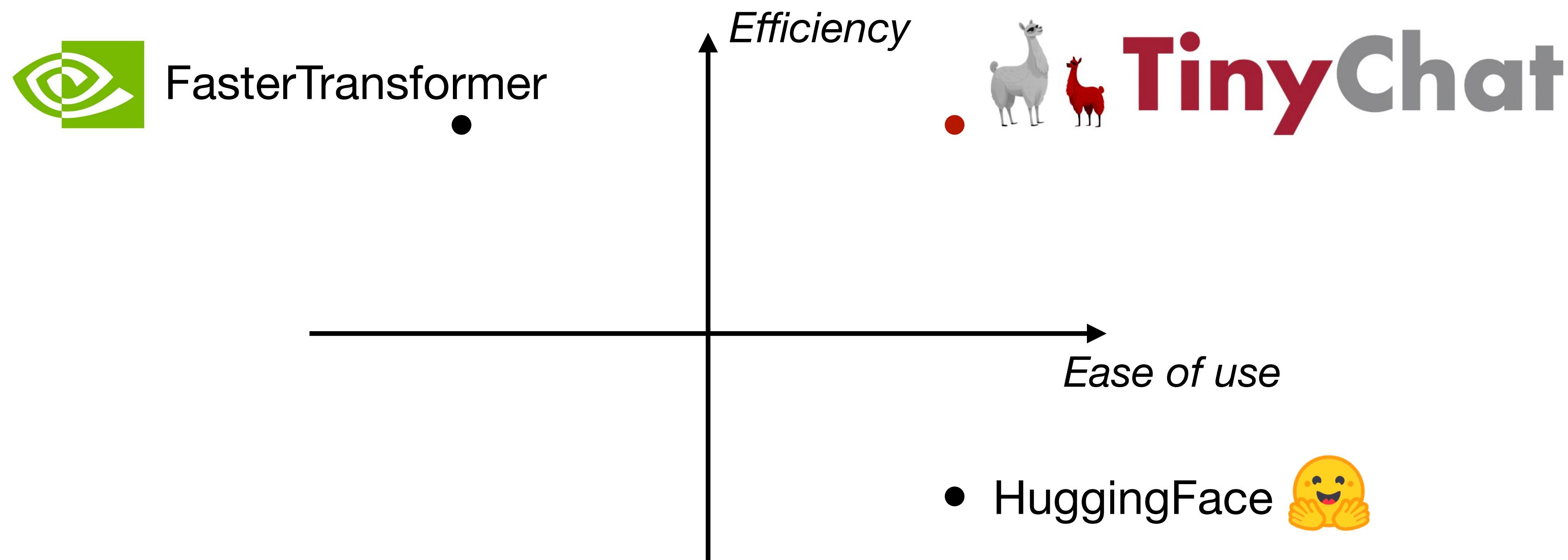
- We need a framework to serve the quantized model to achieve low latency (AWQ only for Linears)
  - HuggingFace: easy to use, but slow
  - FasterTransformer: high efficiency, but harder to use

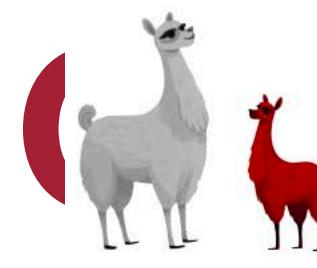


# TinyChat: A Lightweight Serving Infra

Pythonic, lightweight, efficient

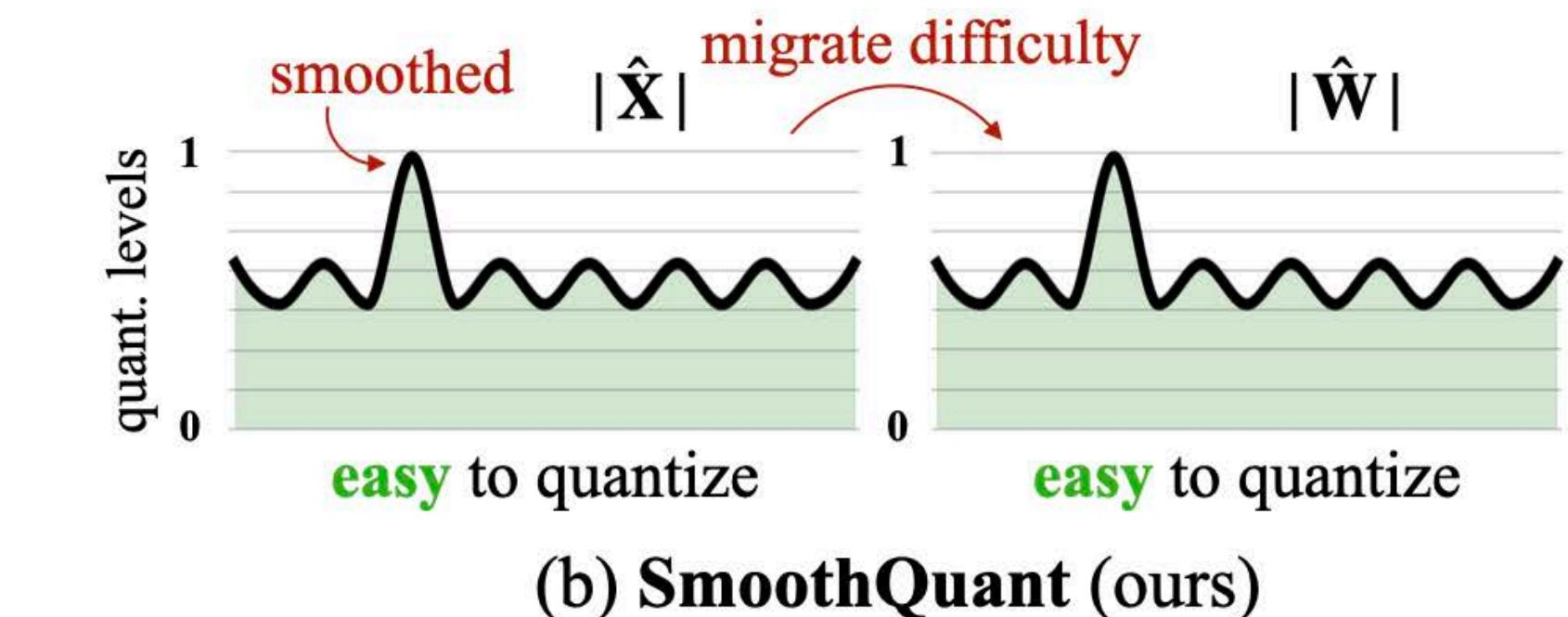
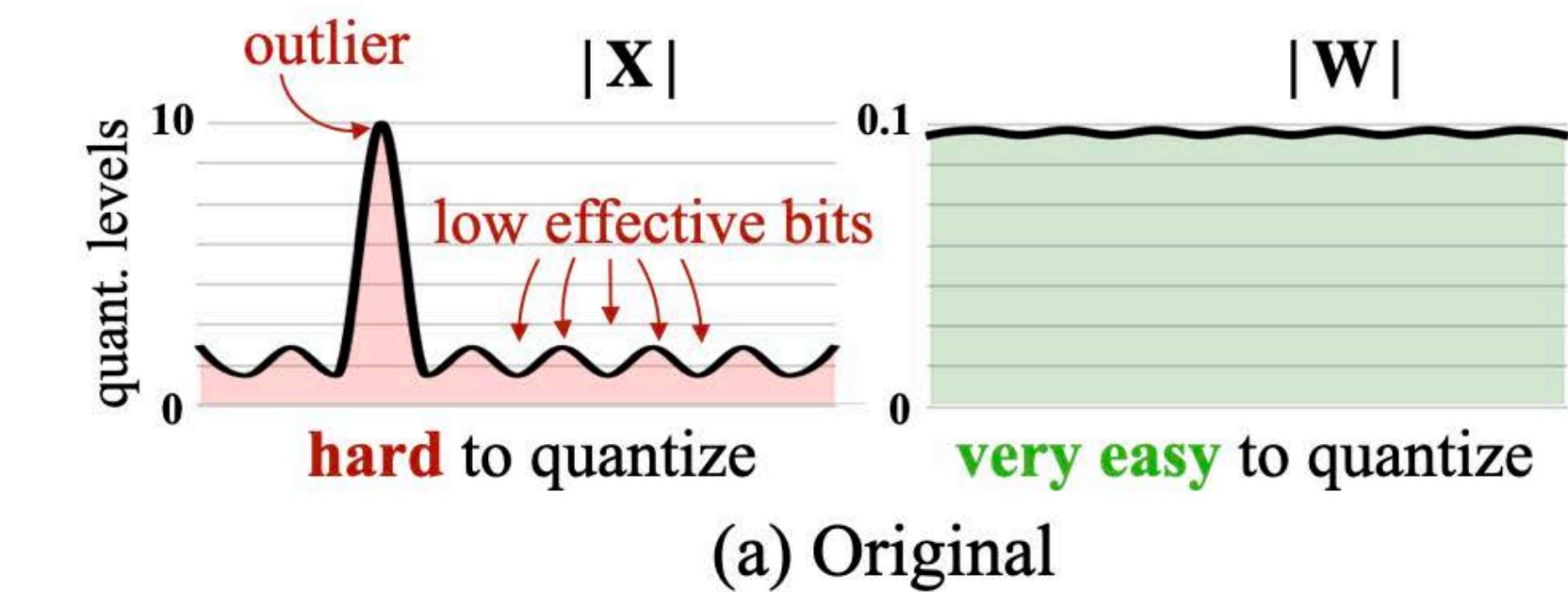
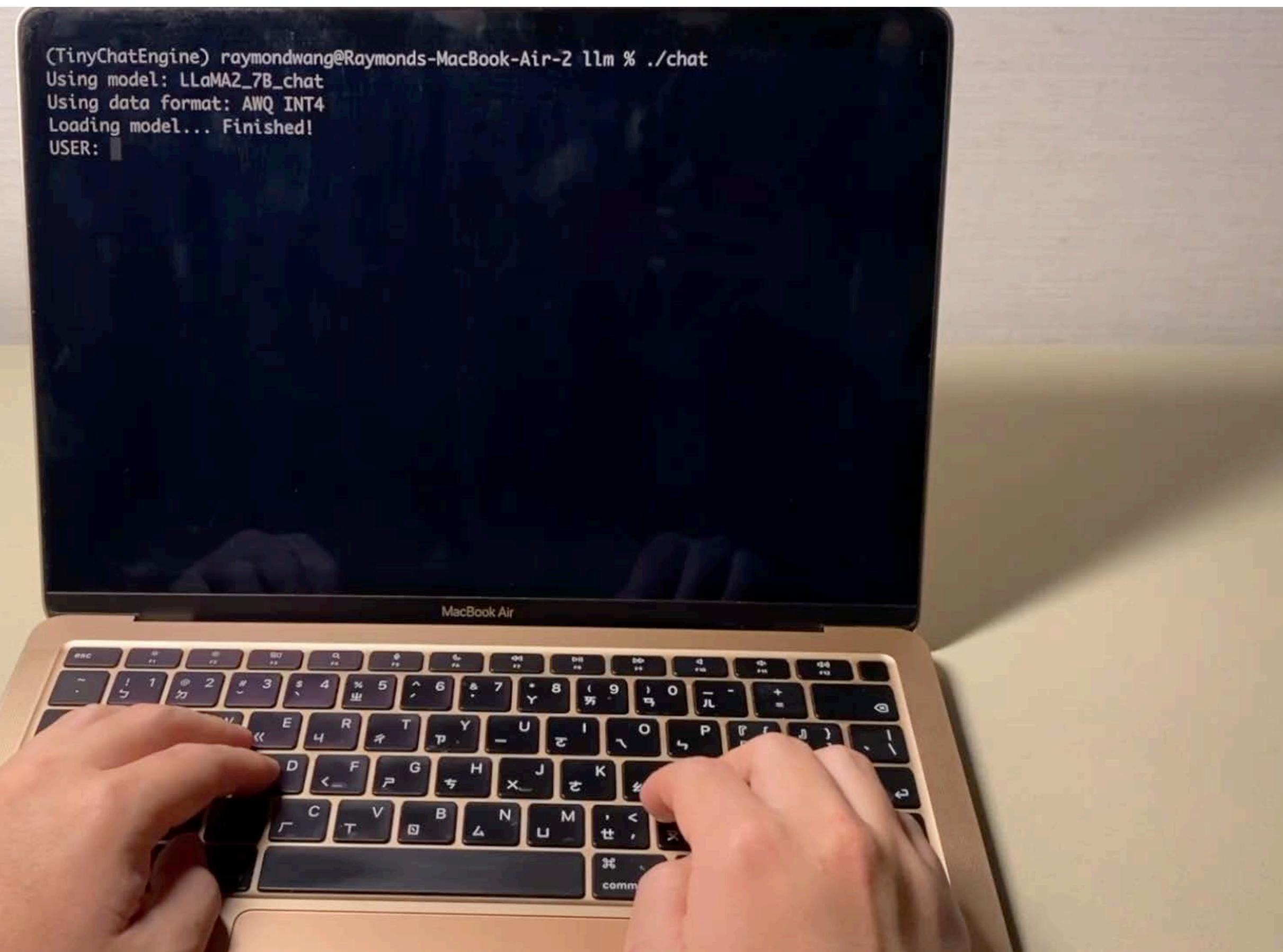
- We need a framework to serve the quantized model to achieve low latency
  - HuggingFace: easy to use, but slow
  - FasterTransformer: high efficiency, but harder to use
- **TinyChat** goals: efficient, lightweight, Python-native (composable with other stacks like vLLM)





# TinyChat M

- We enable edge deployment of LLMs through quantization: SmoothQuant and AWQ
- TinyChatEngine implements the compressed inference, built from C/C++ from scratch, easy to install and migrate to edge platforms



# Improving Multi-Modal LLMs

## Ji's internship project at NVIDIA Research (WIP)

- AWQ makes multi-modal LLM efficient
- But the current open-source multi-modal LLMs have poor visual-language alignment



**Question:** Is the man wearing eyeglasses?

**LLava-7B:** Yes, the man is wearing eyeglasses.

**MiniGPT-4-7B:** Yes, the man is wearing eyeglasses.

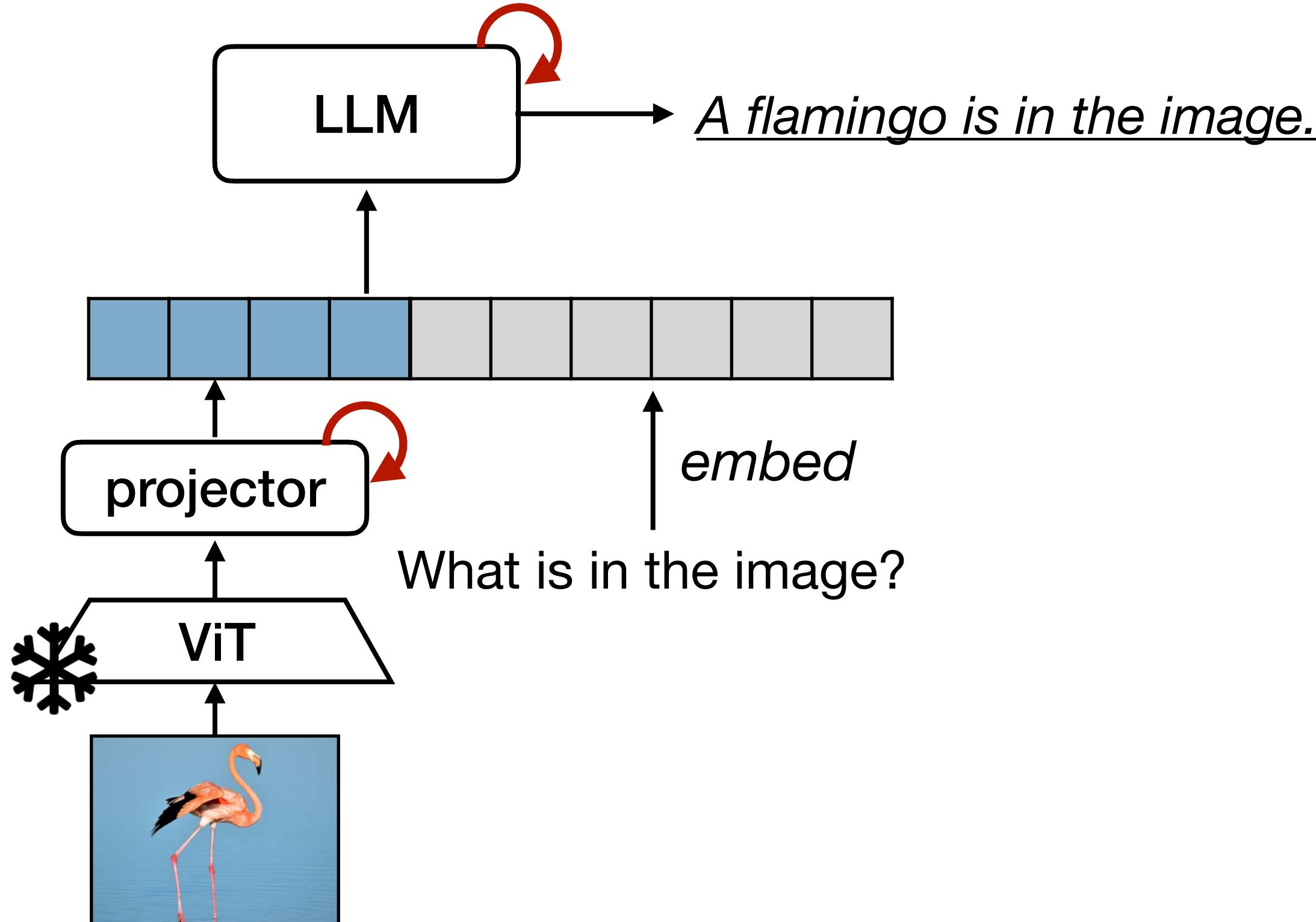
**VPGTrans-7B:** Yes, the man is wearing eyeglasses.

**LLama-Adapter-V2-7B:** Yes, the man is wearing eyeglasses in the image.

# Improving Multi-Modal LLMs

## Improving PaLM-E alike Vision Language Models (VLM)

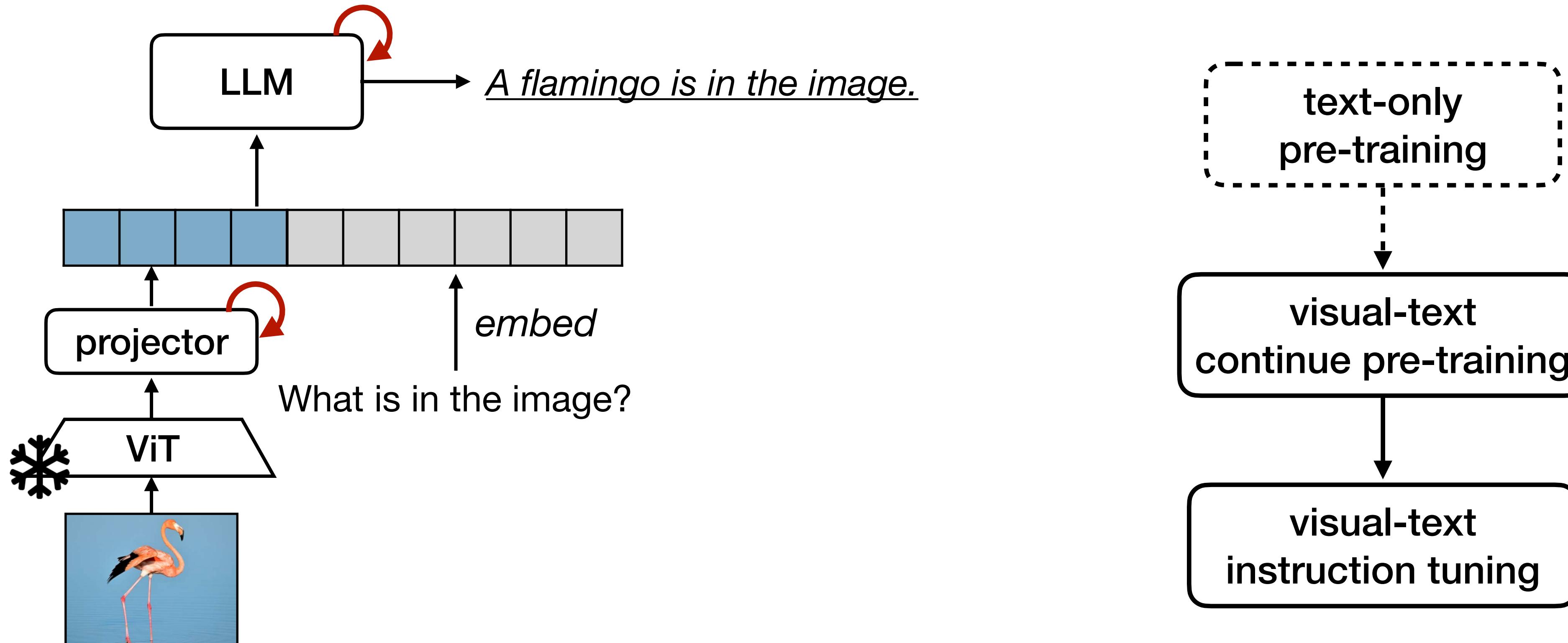
- We study PaLM-E alike architecture due to its **flexibility** for multi-modal input/output
  - *Visual input as a foreign language*
- **Goal:** good performance on both **visual-language** and **text-only**



# Improving Multi-Modal LLMs

## Improving PaLM-E alike Vision Language Models (VLM)

- We study PaLM-E alike architecture due to its **flexibility** for multi-modal input/output
  - *Visual input as a foreign language*
- **Goal:** Multi-modal LLM with good *visual-language* performance and *text-only* performance



# Improving Multi-Modal LLMs

## Qualitative results

- ICL example (counting)



pandas: 3



dogs: 2



Pred: cats: 3

---

*context*

---

*prediction*

# Improving Multi-Modal LLMs

## Qualitative results

- ICL example (world knowledge)



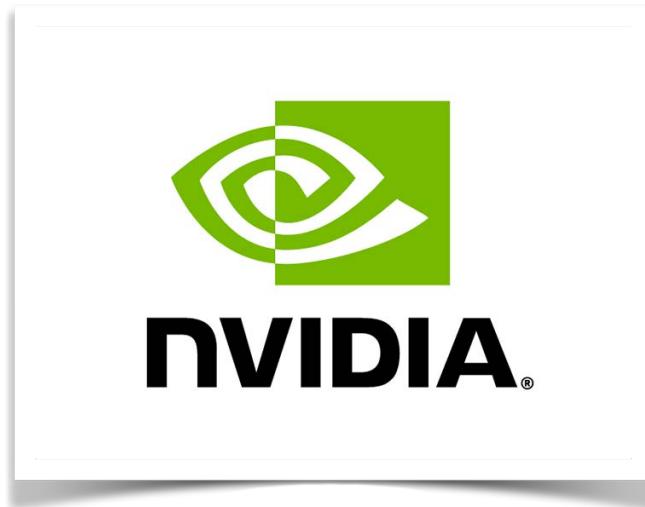
The company is famous for its search engine.



The company is famous for the operating system.



The company is famous for iPhone and Mac.



**Pred:** The company is famous for its graphics processing units (GPUs)

# Improving Multi-Modal LLMs

## Qualitative results

- We can use the model to explain self-driving scenarios (*zero-shot*)



**Q:** Can the vehicle proceed through the traffic now?

**A:** Based on the image, the vehicle cannot proceed through the traffic yet. There are multiple people and bicycles in the crosswalk, and the traffic light is red. The vehicle must wait for the traffic light to turn green before proceeding.

# Improving Multi-Modal LLMs

## Qualitative results

- Explaining an NVIDIA meme



**Q:** Can you explain the meme?

**A:** The meme depicts a man's reaction to the price of a computer graphics card. In the first image, the man is smiling and appears excited about the product. In the second image, he is shocked and disappointed by the high price of the graphics card, which is \$1,200. The meme is a playful representation of the contrast between the man's initial enthusiasm and his subsequent disappointment upon learning the cost of the product.

# SpAtten: Transformer with Sparse Attention

Token Pruning: not every token is equally important. Progressively prune them.

As a visual treat, the film is almost perfect.

11 Tokens  $\downarrow$  8 Heads

BERT Layer 1 (100% Computation & Memory Access)

As treat, film perfect.

6 Tokens  $\downarrow$  5 Heads

Layer 2 (34%)

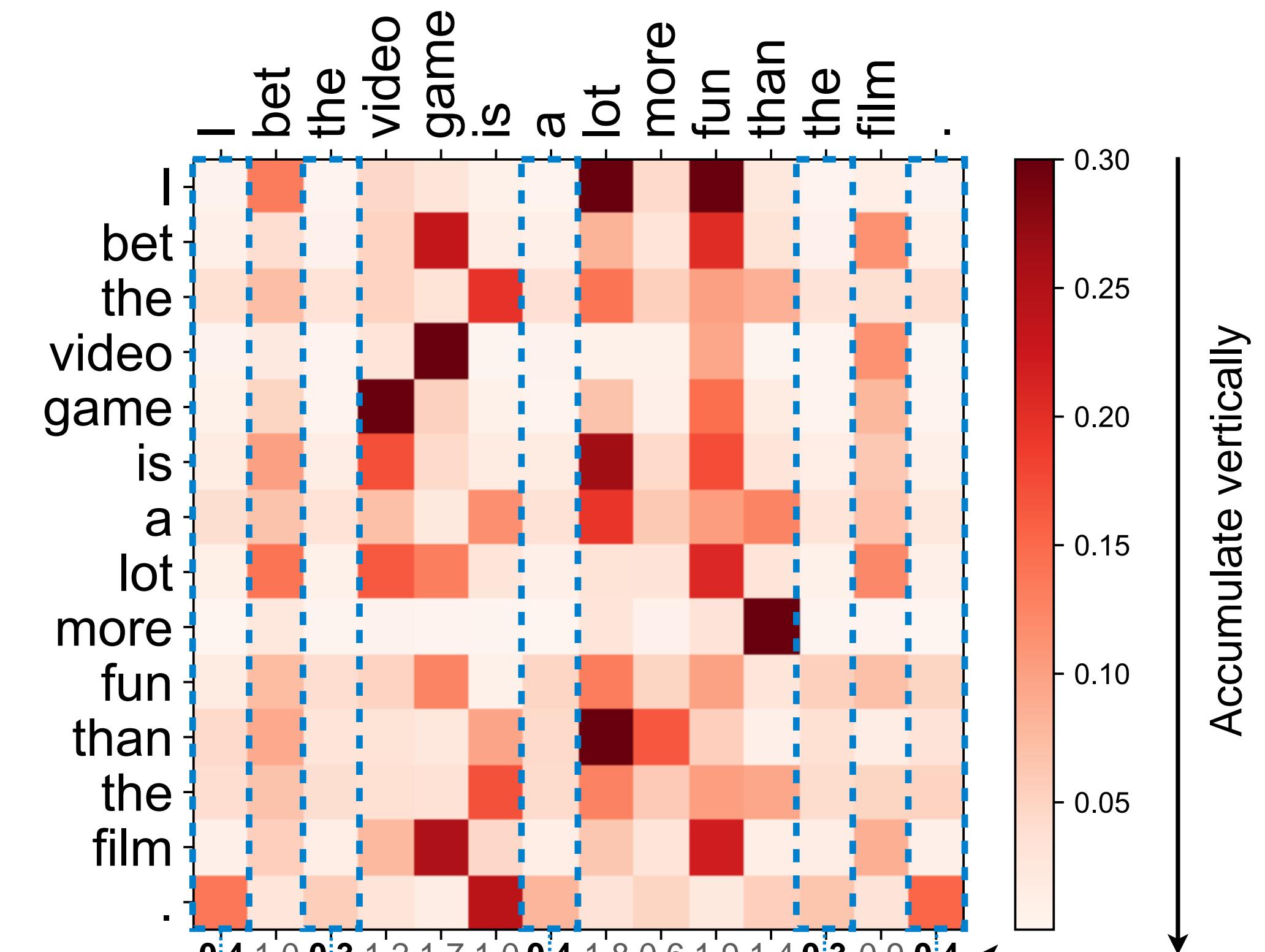
film perfect

2 Tokens  $\downarrow$  4 Heads

Layer 3 (9%)

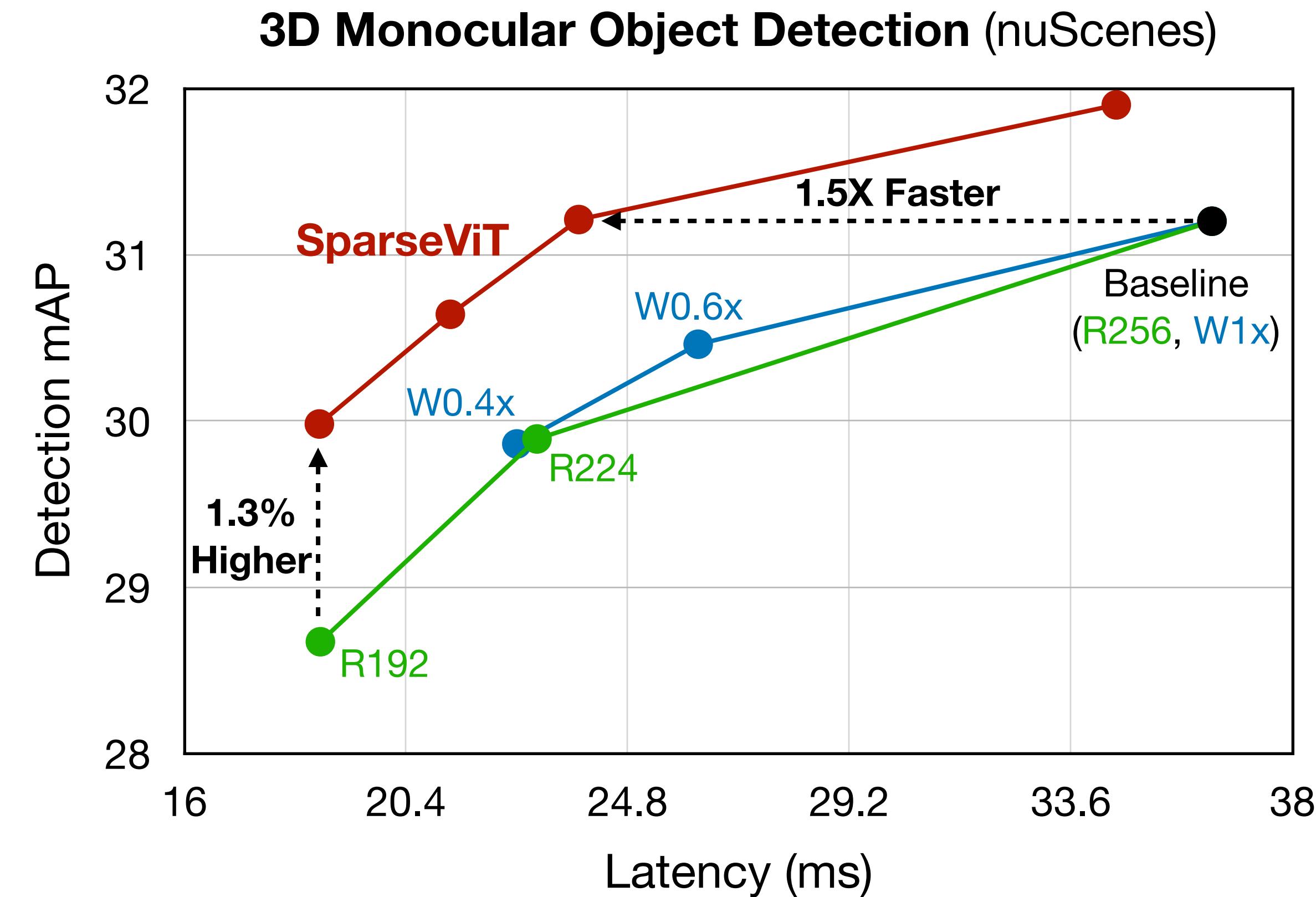
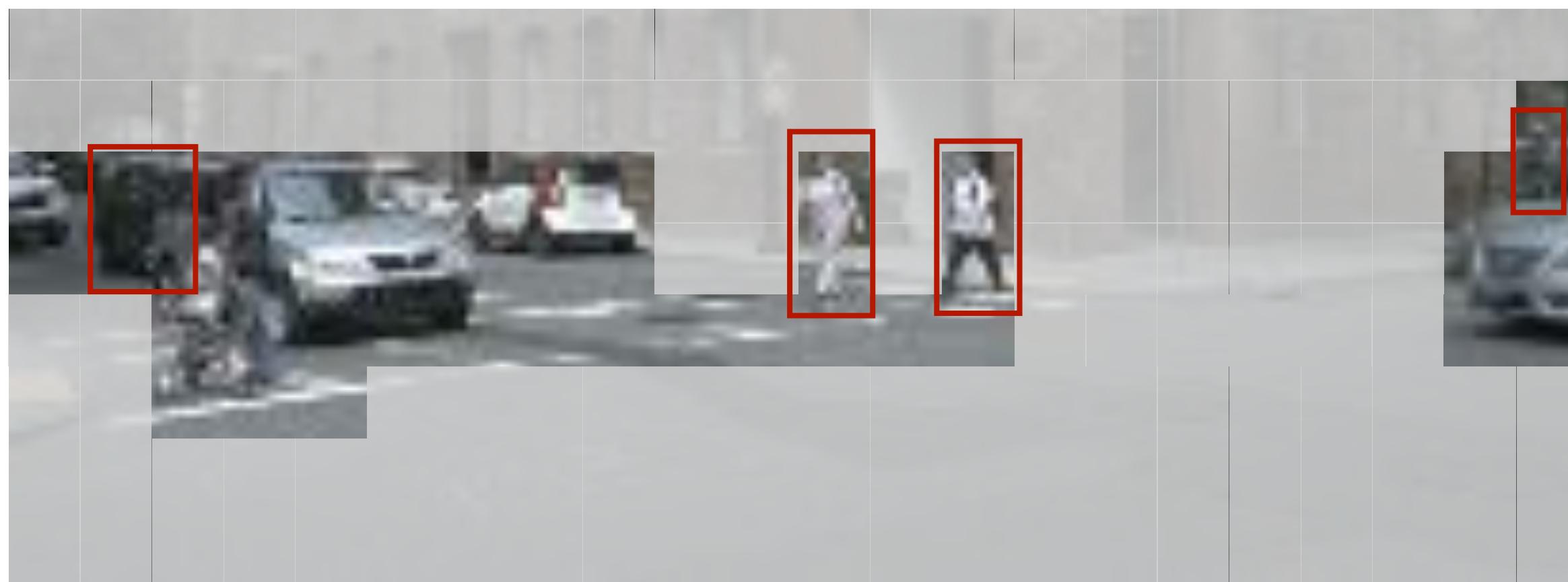
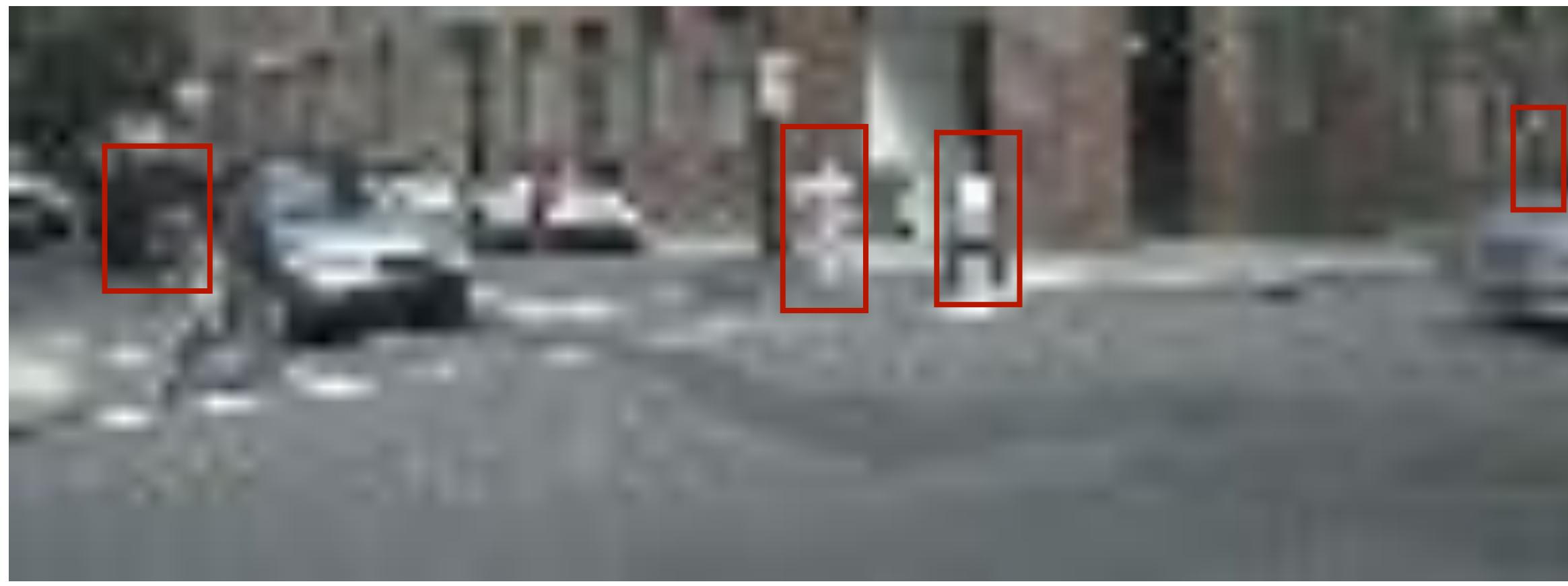
Sentiment Classification: Positive 

Remove redundant token and head according to cumulative importance



# SparseViT: Token Pruning for ViTs

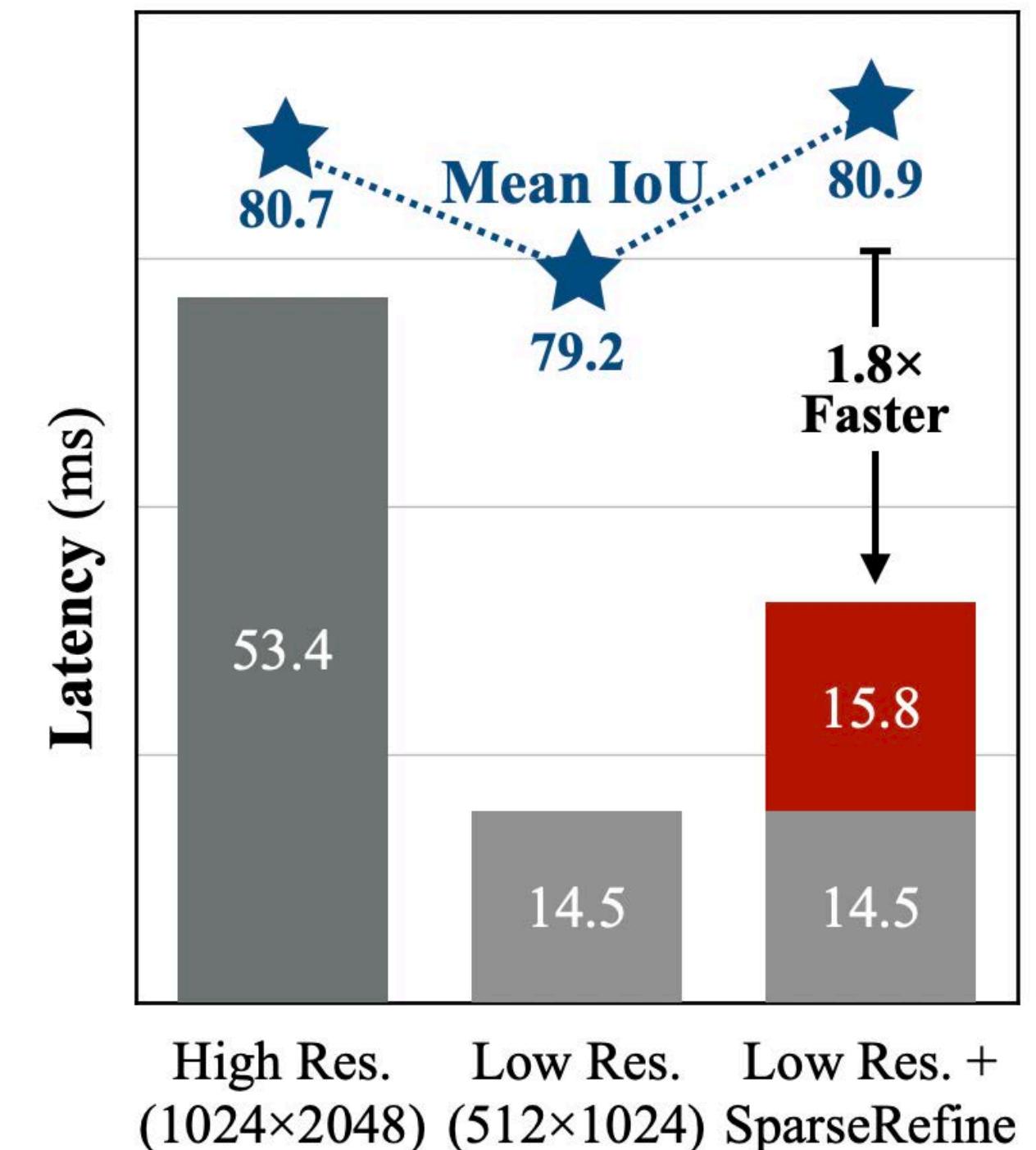
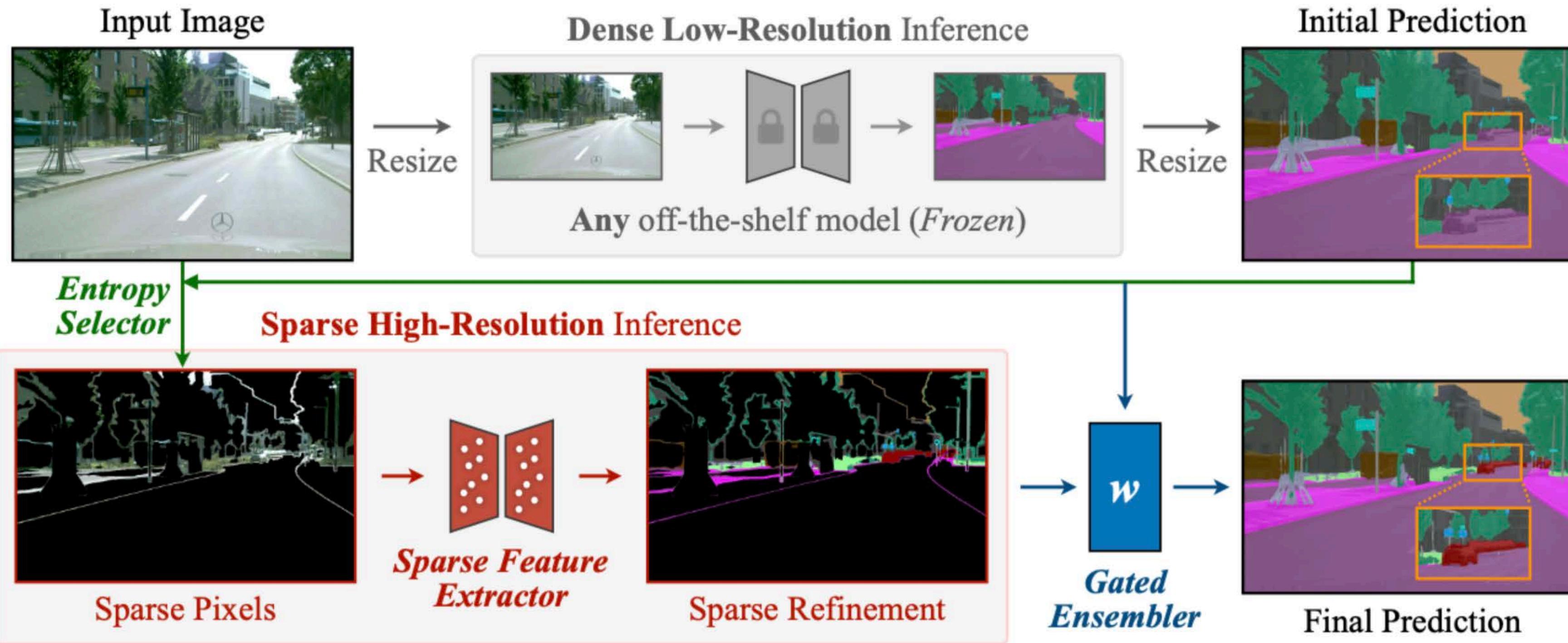
Sparse high-resolution features are better than dense low-resolution ones



- 1.5X faster than the baseline without loss of accuracy.
- 1.3% higher mAP than the baseline with reduced resolution.

# SparseRefine: Pixel Pruning for CNN

sparse high-resolution + dense low-resolution



## Enabled by TorchSparse

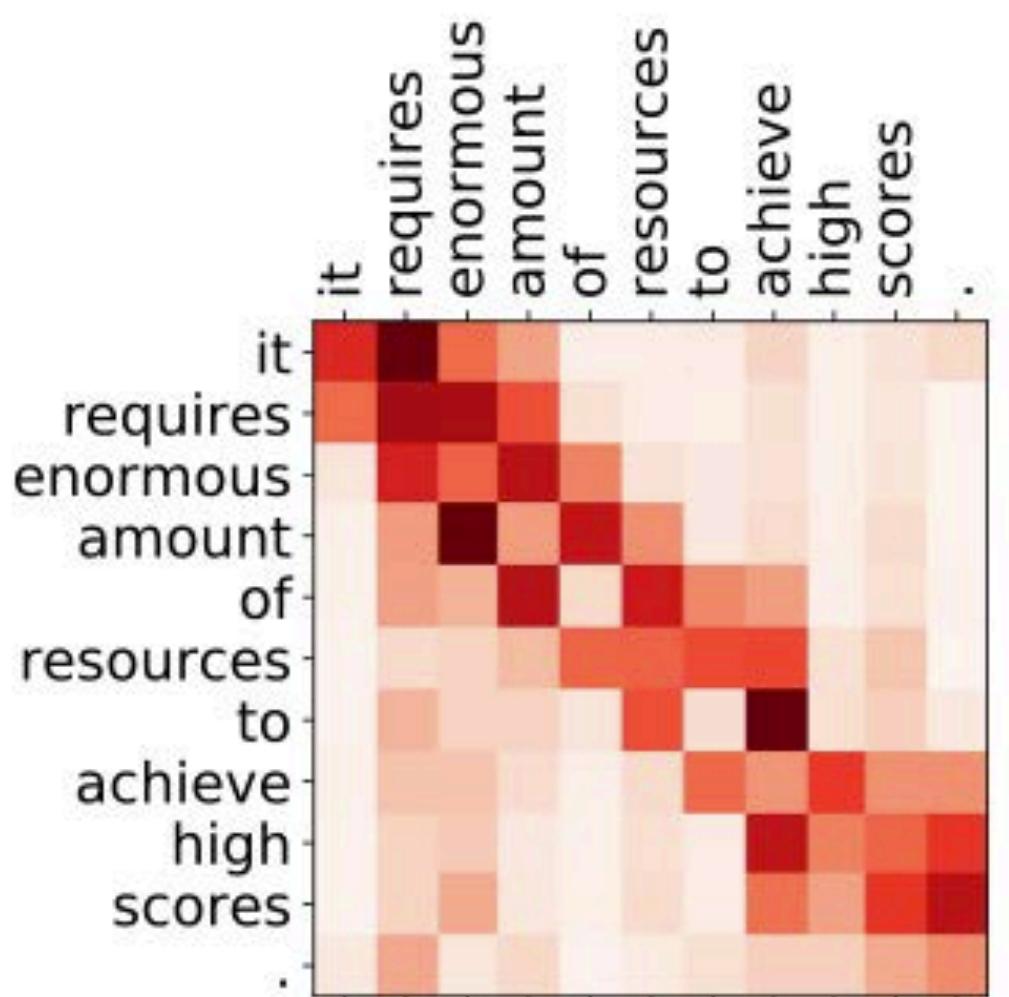
Tang et al. [MICRO'23]

Tang et al. [MLSys'22]

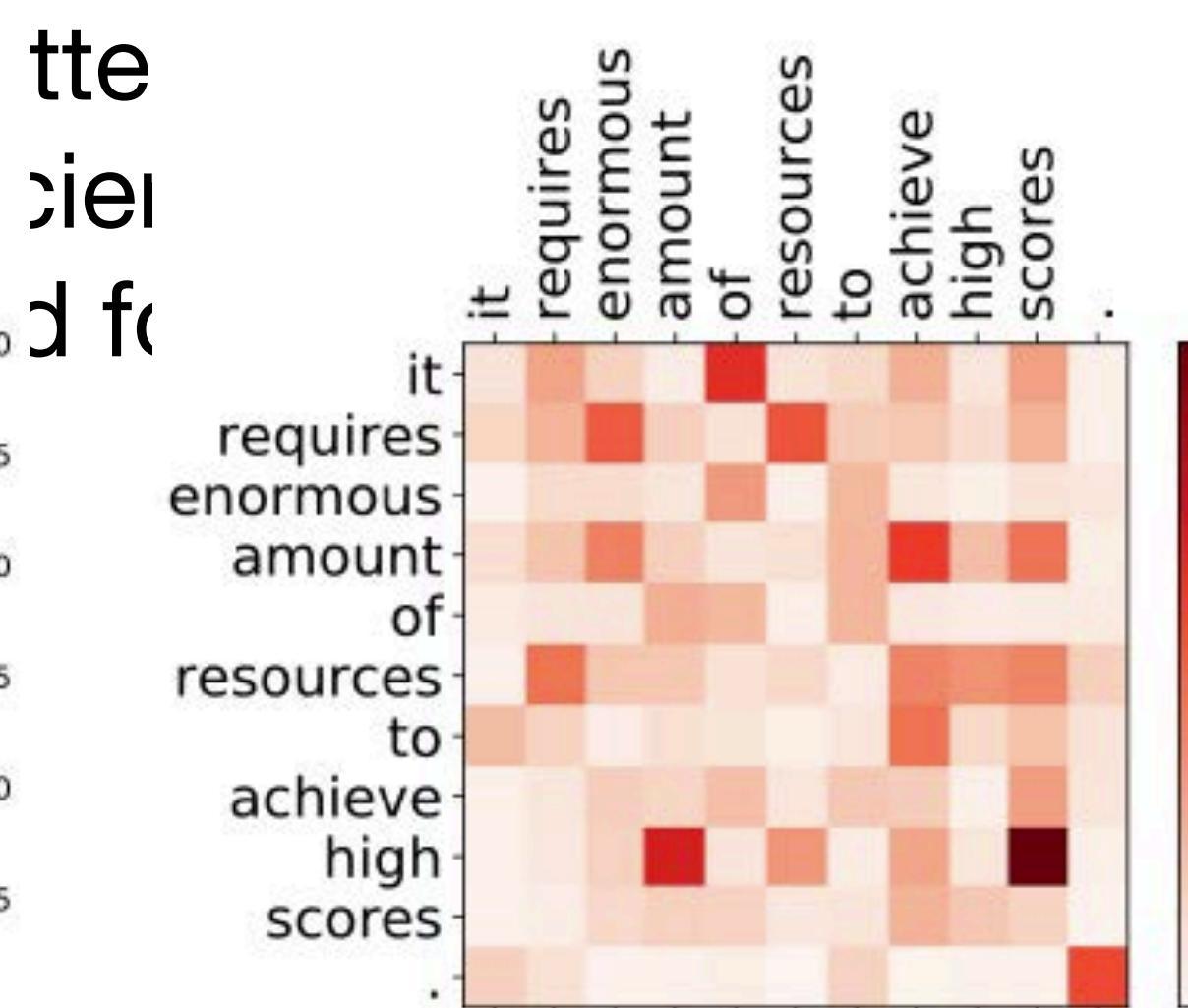
# Lite Transformer

## Specialization: Convolution (local feature) + Attention (global feature)

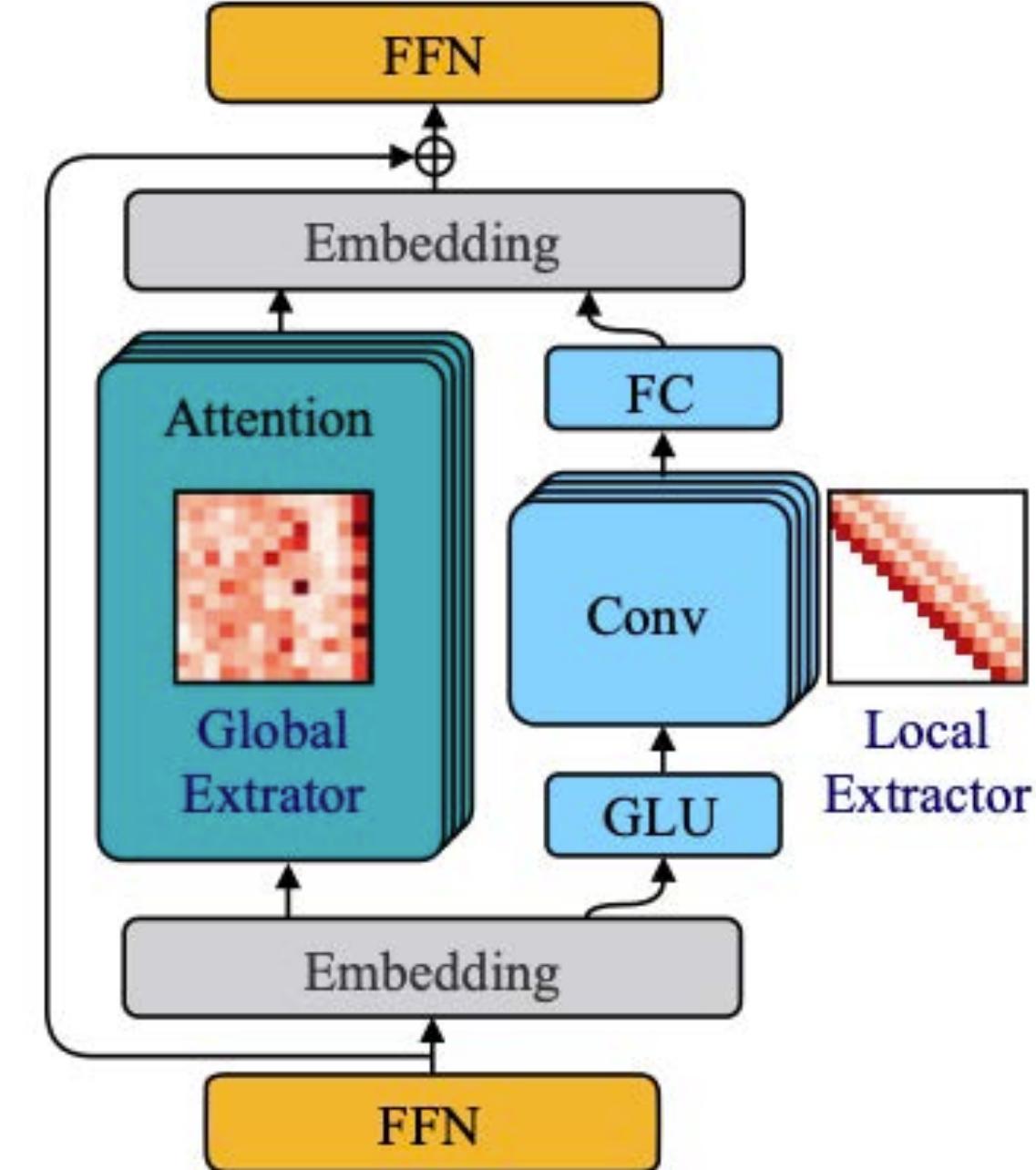
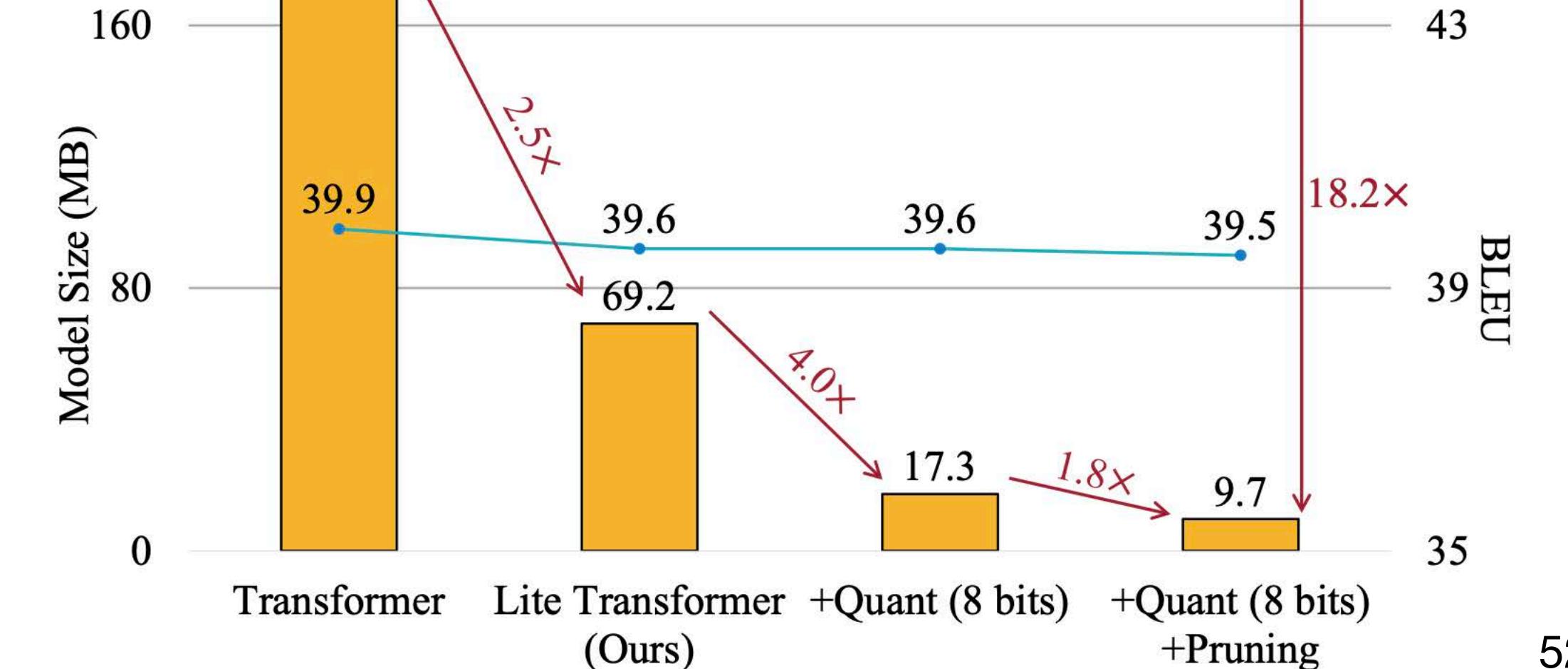
**Original Attention**  
(Too much emphasize on local feature extraction)



**Attention in LSRA**  
(Dedicated for global feature extraction)

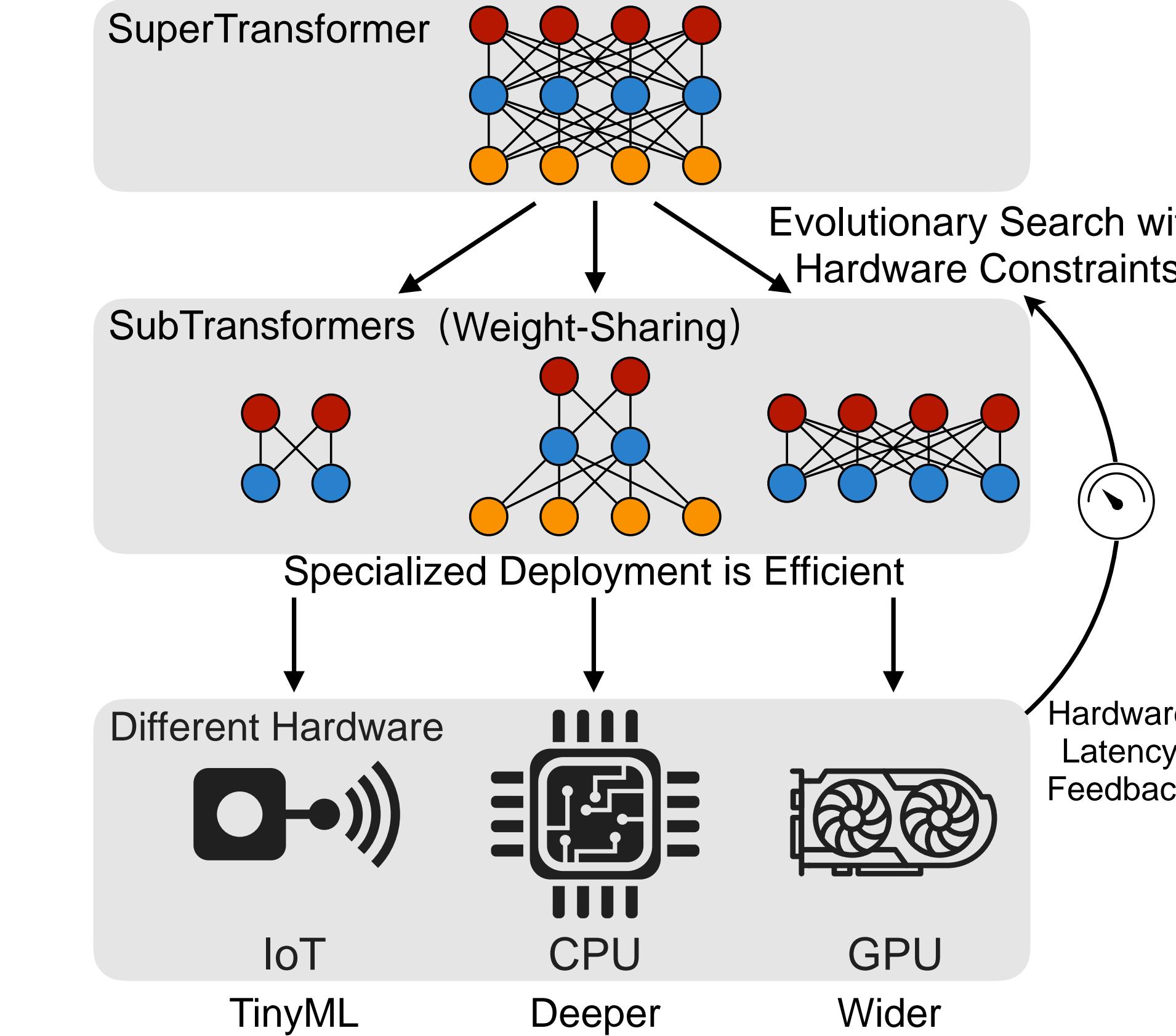


short-range) features.  
feature extraction.

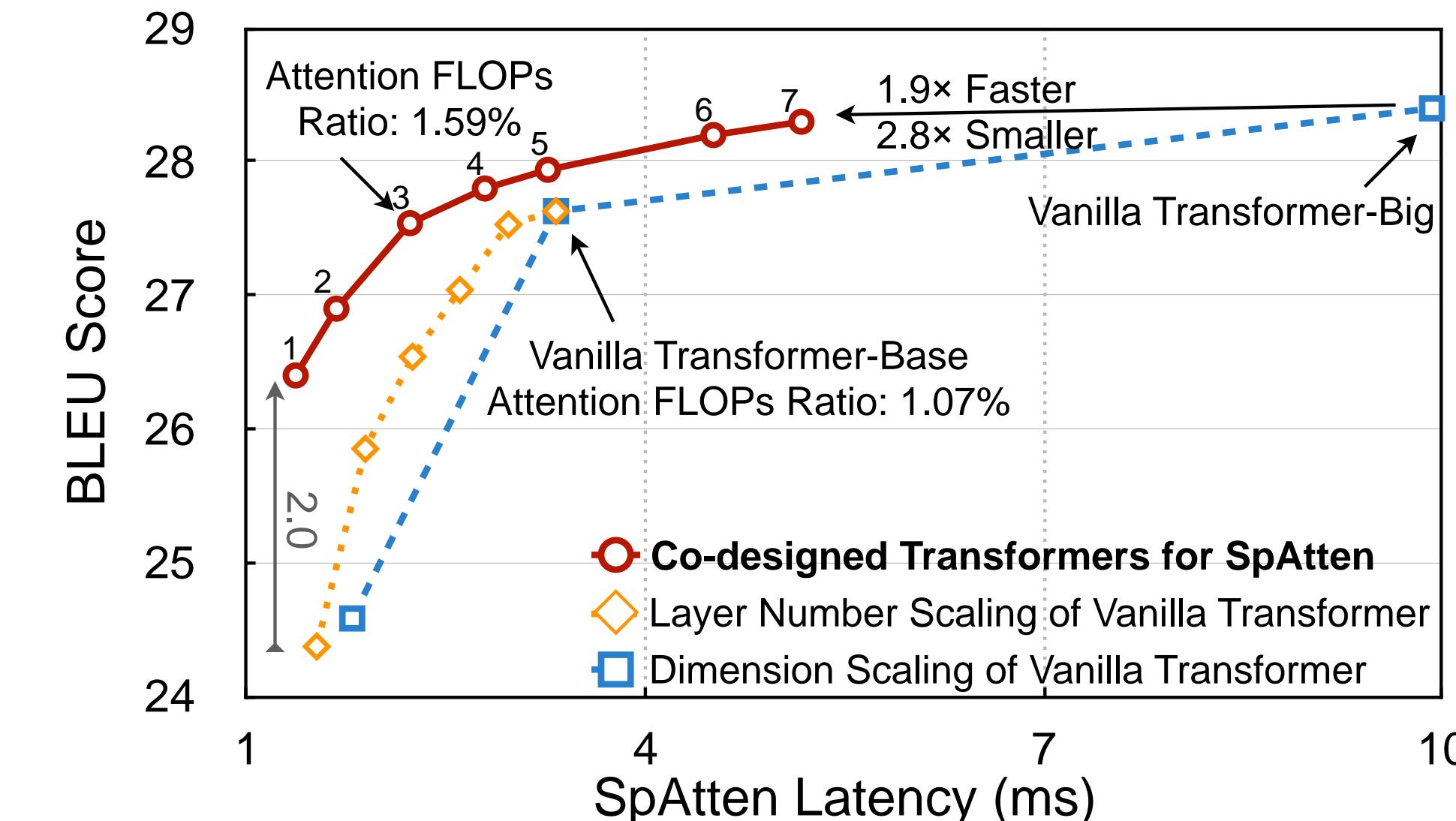
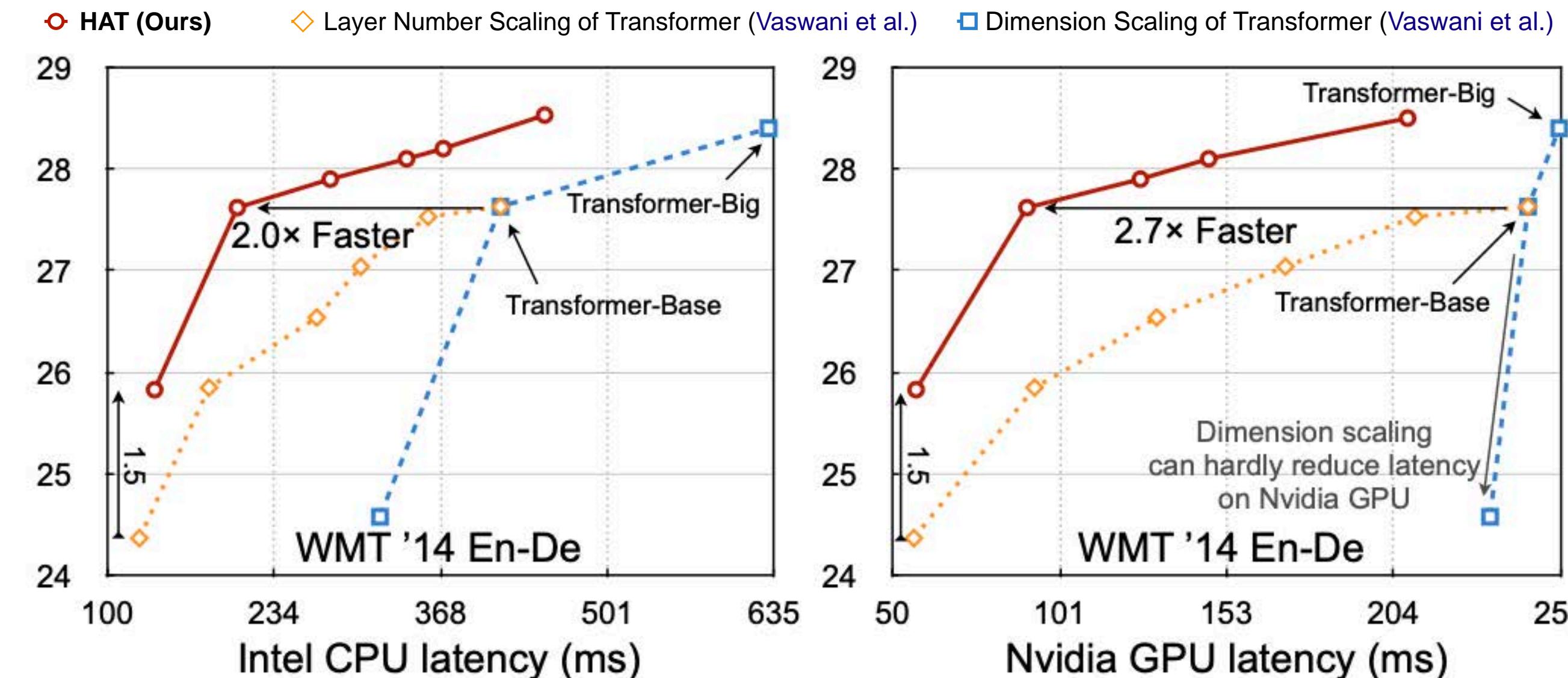


# HAT: Hardware Aware Transformer

Specialization: design different transformer architecture tailored for different hardware



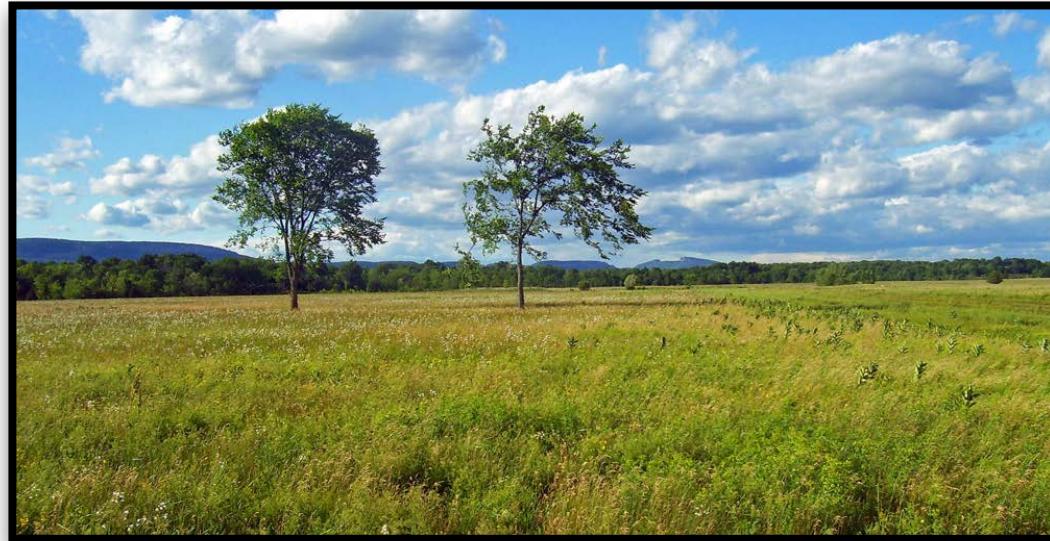
Search for most efficient Transformer model with hardware feedback



# Spatially Sparse Inference for Diffusion Models

## Qualitative Results of SIGE on Stable Diffusion

*A photograph of a horse on a grassland.*



Original



11.6% Masked



Stable Diffusion:  
1855GMACs 369ms



Ours:  
514G (3.6×) 95.0ms (3.9×)

Image Inpainting

Latency Measured on NVIDIA RTX 3090



*A fantasy beach landscape, trending on artstation.*



Original



2.9% Edited



Stable Diffusion+SDEdit:  
1855GMACs 369ms



Ours:  
353G (5.3×) 76.4ms (4.8×)

Image Editing

# Fast Composer

To run the demo, you should:

1. Upload your images. The order of image1 and image2 needs to match the order of the subjects in the prompt. You only need 1 image for single subject generation.
2. Input proper text prompts, such as "A woman img and a man img in the snow" or "A painting of a man img in the style of Van Gogh", where "img" specifies the token you want to augment and comes after the word.
3. Click the Run button. You can also adjust the hyperparameters to improve the results. Look at the job status to see if there are any errors with your input.

Image 1

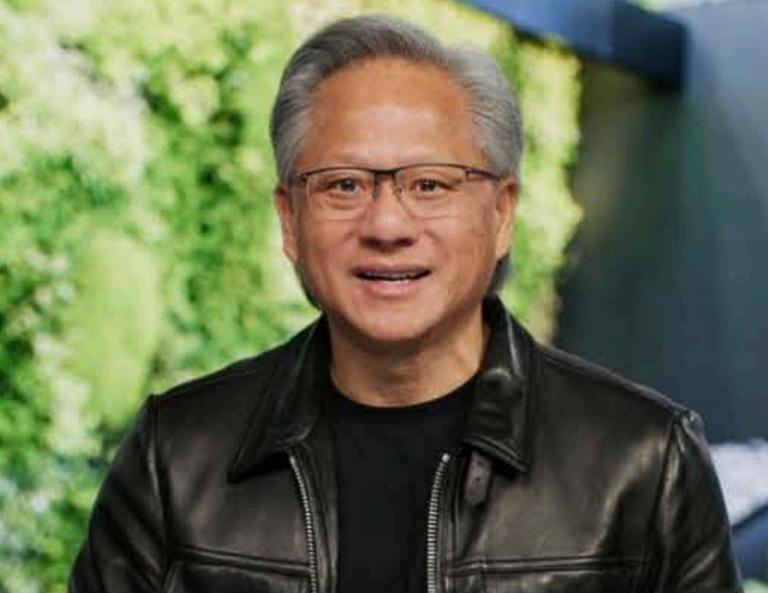


Image 2



Upload the image for your subject

Prompt

Use "img" to specify the word you want to augment.

A man img and a woman img singing together

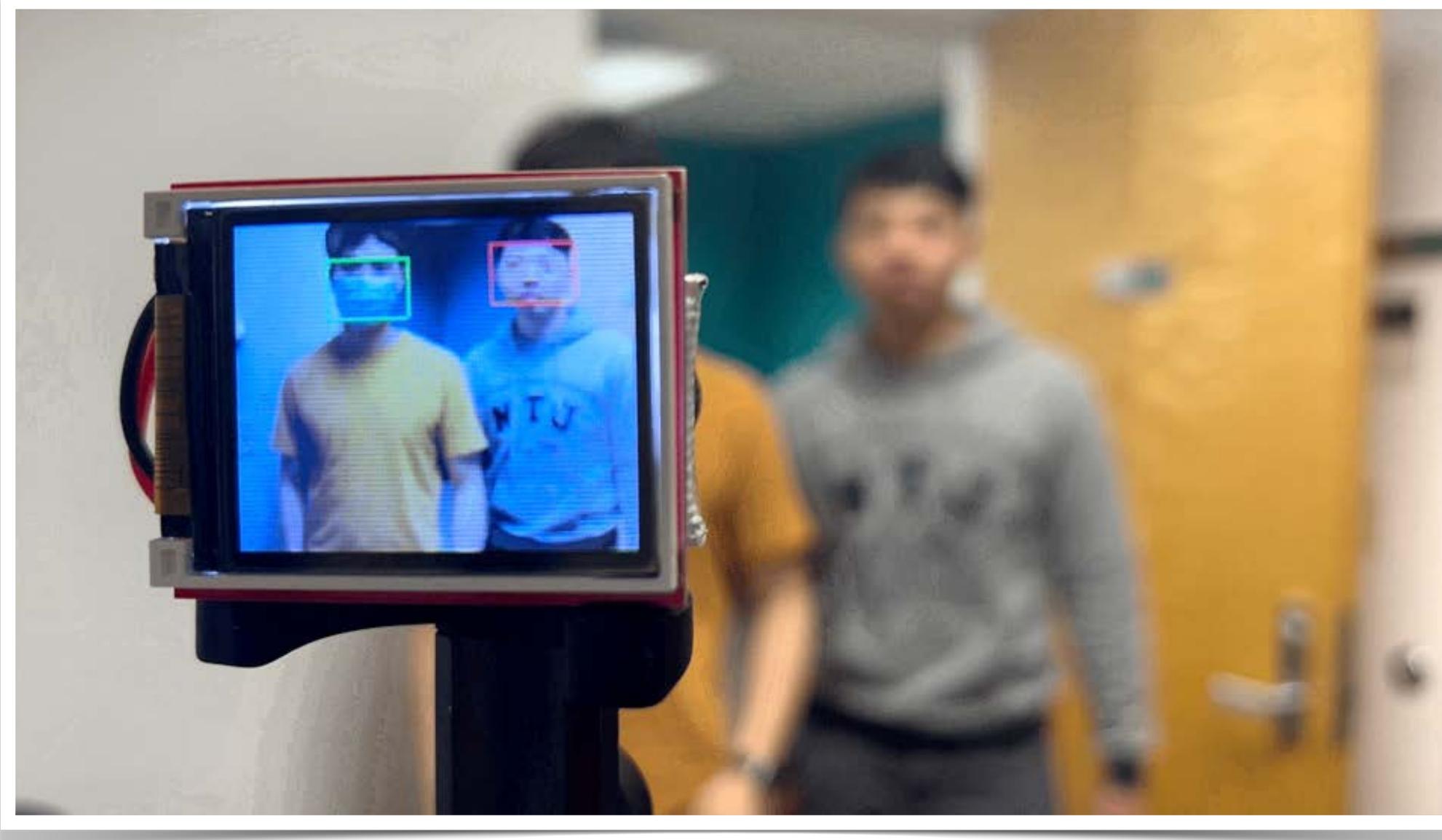
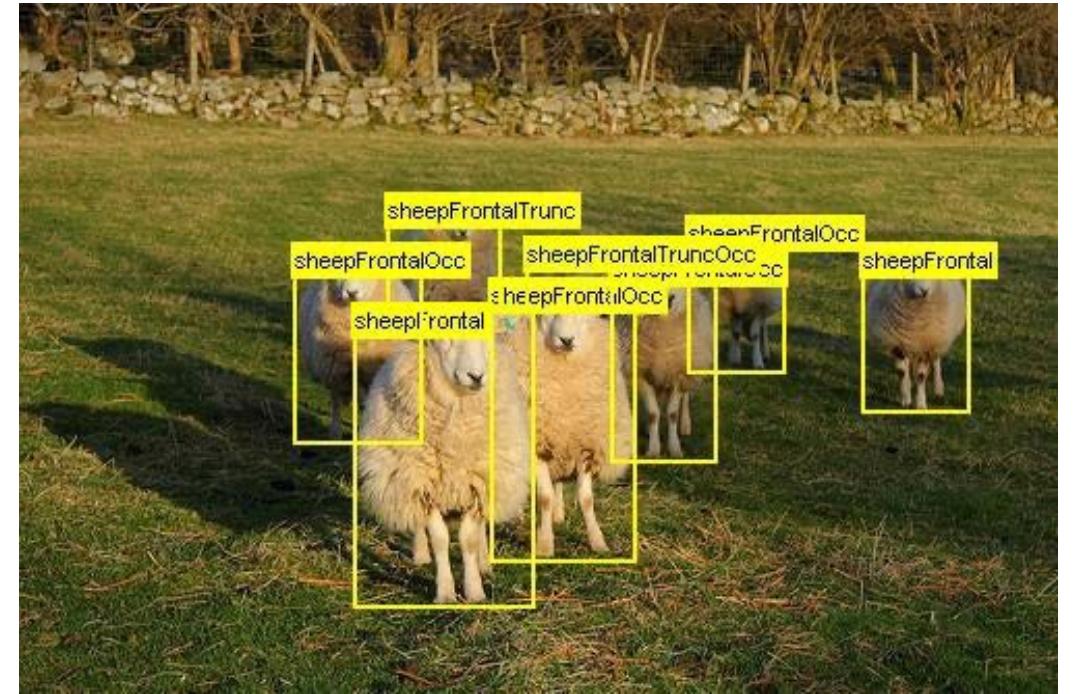
Job Status

run successfully

# MCUNet for TinyML

Advancing object detection by allowing a larger resolution

- Resolution is more important for detection than classification
- Our method significantly improves objection detection by double digits



Face/mask detection



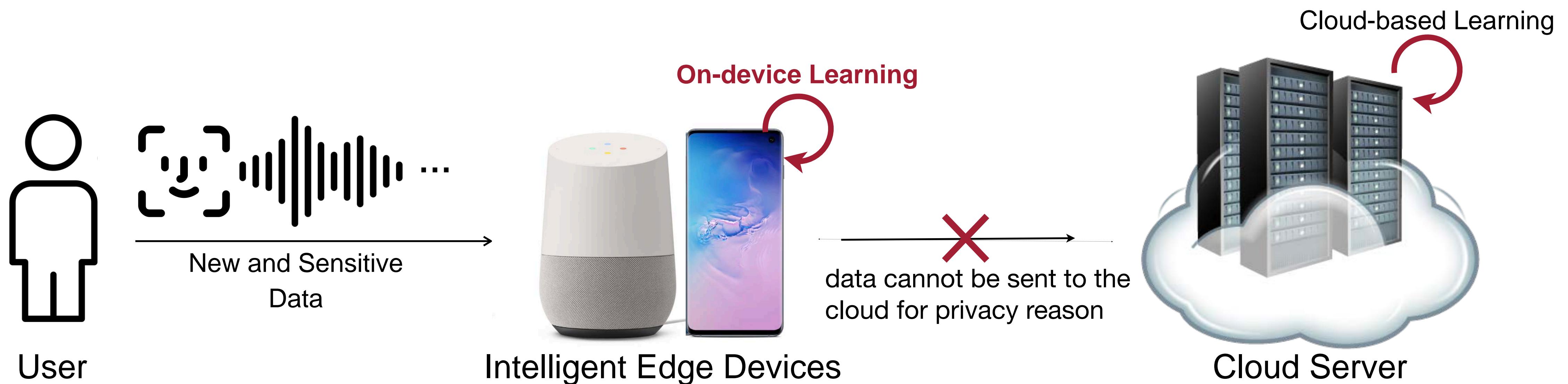
Person detection

**OpenMV Cam: 512KB SRAM + 2MB Flash**

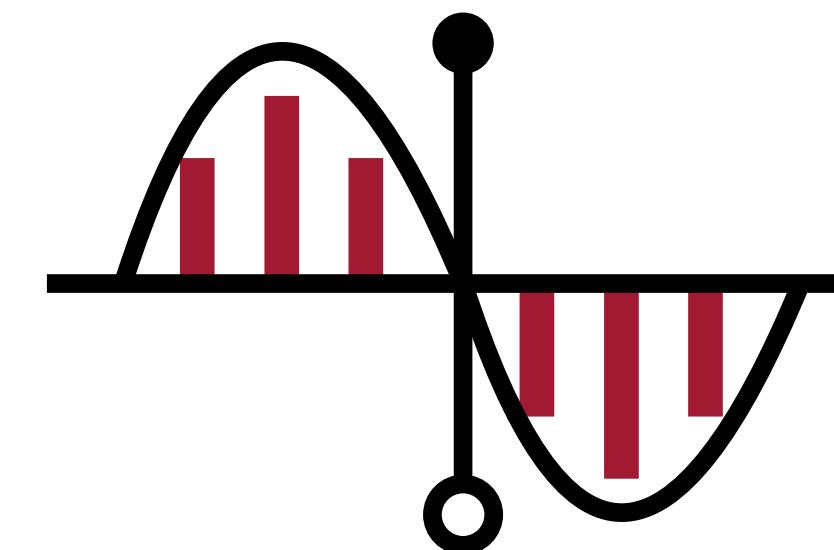
# Can We Learn on Edge?

AI systems need to continually adapt to new data collected from the sensors

- On-device learning: **better privacy, lower cost, customization, life-long learning**
- Training is more **expensive** than inference, hard to fit edge hardware (limited memory)



# On-Device Training Under 256KB Memory

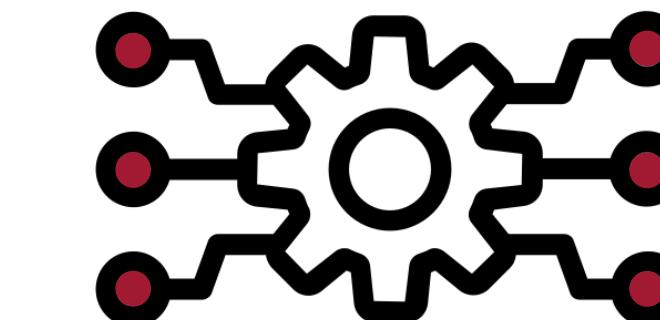


## 1. Quantization-aware scaling

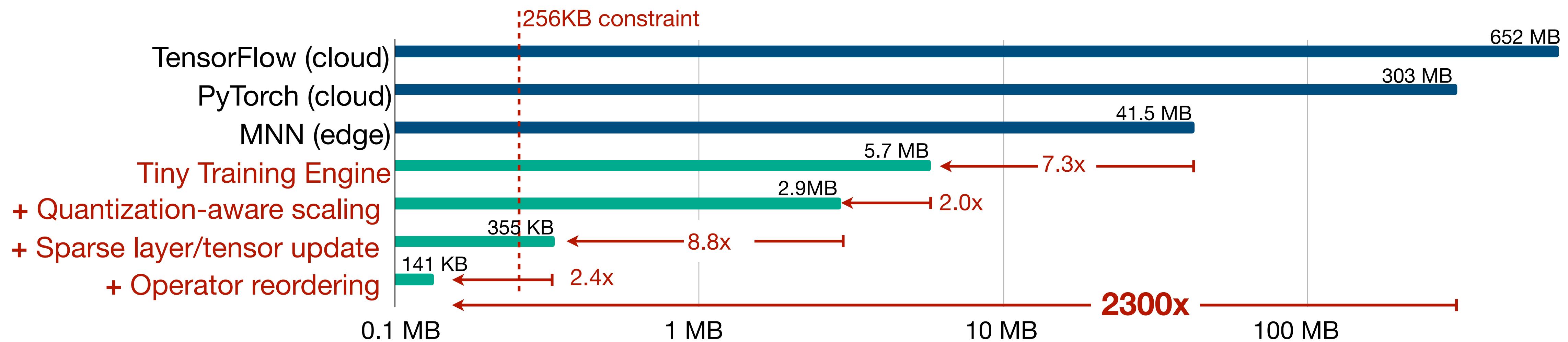
$$\tilde{G}_{\bar{W}} = G_{\bar{W}} \cdot s_{\bar{W}}^{-2}$$



## 2. Sparse layer/tensor update

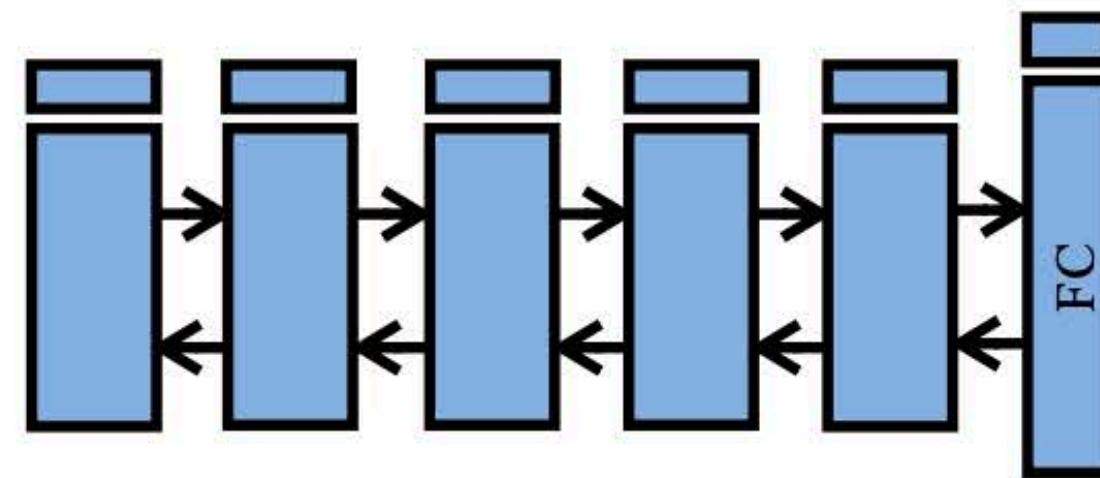


## 3. Tiny Training Engine

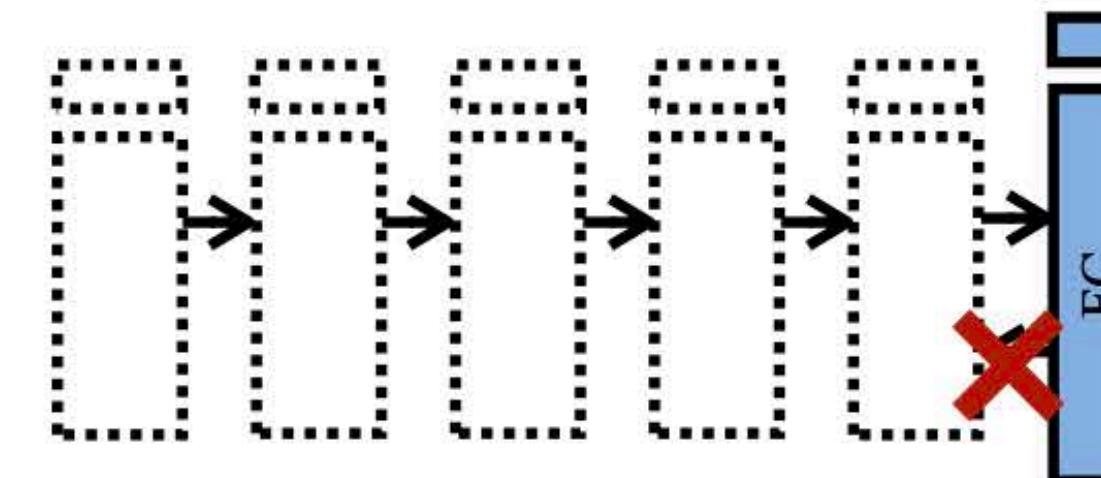


# PockEngine: Sparse and Efficient Fine-tuning in a Pocket

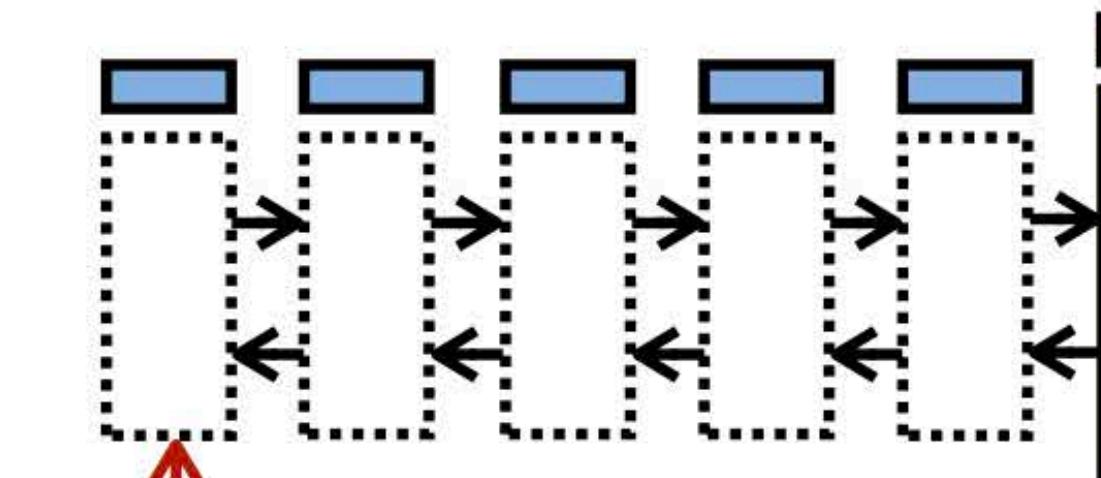
Finetune Llama-7B on edge device (to appear at MICRO'23)



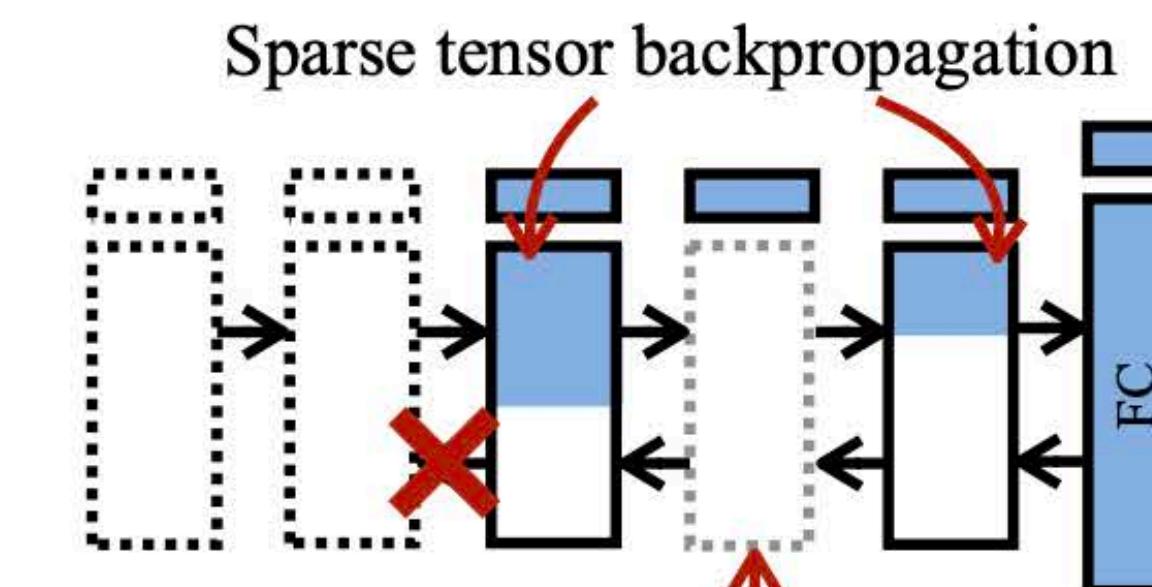
(a) Full BP



(b) Last-only BP



(c) Bias-only BP



Sparse tensor backpropagation  
Sparse layer backpropagation  
(d) Sparse BP

Framework	Method	Iteration Latency (↓)	Alpaca-Eval Winrate(↑)
PyTorch	FT-Full	7.7s	44.1%
PyTorch	LoRA (rank=8)	7.3s	43.1%
PockEngine	FT-Full	1.8s	43.7%
PockEngine	Sparse	0.9s	43.1%

**Input:** Micro was running virtually online in 2021, and held physically at Chicago in 2022 and will soon be hosted in Toronto in 2023. Now is the year of 2023, where was Micro hosted last year?

**LlamaV2-7B-original:** Micro was hosted virtually in 2021

**LlamaV2-7B-tuned (Full-BP):** Micro was hosted physically in Chicago in 2022.

**LlamaV2-7B-tuned (Sparse-BP):** Based on the information provided, it seems that Micro was hosted physically at Chicago in 2022.

Example 5. Inferring Information Based on Context.

# PockEngine: Sparse and Efficient Fine-tuning in a Pocket

## Finetune Llama-7B on edge device (to appear at MICRO'23)

**Input:** Lee mows one lawn and charges \$33. Last week he mowed 16 lawns and three customers each gave him a \$10 tip. How many dollars did Lee earn mowing lawns last week?

**LlamaV2-7B-original:** Lee earned \$660 from mowing lawns last week.

**LlamaV2-7B-tuned (Full-BP):** Lee earned  $\$33 \times 16 = \$528$  from mowing lawns. His total earnings from the 16 customers who gave him a \$10 tip is  $\$10 \times 3 = \$30$ . Therefore, Lee earned a total of  $\$528 + \$30 = \$558$  last week mowing lawns.

**LlamaV2-7B-tuned (Sparse-BP):** To determine how much Lee earned mowing lawns last week, we need to calculate the total amount of money he earned from mowing lawns and tips. Lee mowed one lawn and charged \$33, so he earned \$33 from that lawn. Last week, Lee mowed 16 lawns, and three customers each gave him a \$10 tip, so he earned  $\$10 \times 3 = \$30$  in tips from those lawns and earned \$33 from mowing one lawn and total 16 lawns = \$528. Therefore, Lee earned \$558 last week from mowing lawns and receiving tips.

Example 6. Math Problem Solving

mit-han-lab / mcunet Public

Notific

Code Issues 6 Pull requests Actions Projects Security Insights

Product Team Enterprise Explore Marketplace Pricing

master 2 branches 0 tags

README.md

## MCUNet: Tiny Deep

This is the official implementation of the

[website](#) | [paper](#) | [paper \(v2\)](#) | [de](#)



## TinyEngine

This is the official implementation of TinyEngine, a co-design framework for tiny deep learning on microcontrollers. TinyEngine is a part of MCUNet, a tight memory budgets.

The MCUNet and TinyNAS repo is [here](#).

[MCUNetV1](#) | [MCUNetV2](#) | [MCUNetV3](#)

Sign up here to get updates!

<https://forms.gle/UW1uUmnfk1k6UJPPA>

# Open Source



mit-han-lab / tinyengine Public

Notifi

Code Issues 1 Pull requests Actions Projects Wiki Security Insights Settings

master 1 branch 0 tags

README.md

main 1 branch 0 tags

Go to file Add file Code

 Lyken17	Merge branch 'main' of <a href="https://github.com/mit-han...">https://github.com/mit-han...</a>	f8dfb50	yesterday	4 commits
 algorithm	prepare open source		2 days ago	
 compilation	prepare open source		2 days ago	
 figures	refine qas_accuracy figure		yesterday	
 .gitignore	prepare open source		2 days ago	
 .gitmodules	prepare open source		2 days ago	
 LICENSE	prepare open source		2 days ago	
 README.md	minor update		yesterday	
 assets	prepare open source		2 days ago	
 configs	prepare open source		2 days ago	

README.md

## On-Device Training Under 256KB Memory

### About

On-Device Training Under 256KB Memory [NeurIPS'22]

[tinytraining.mit.edu](#)

edge-ai on-device-training  
learning-on-the-edge

Readme

MIT license

65 stars

8 watching

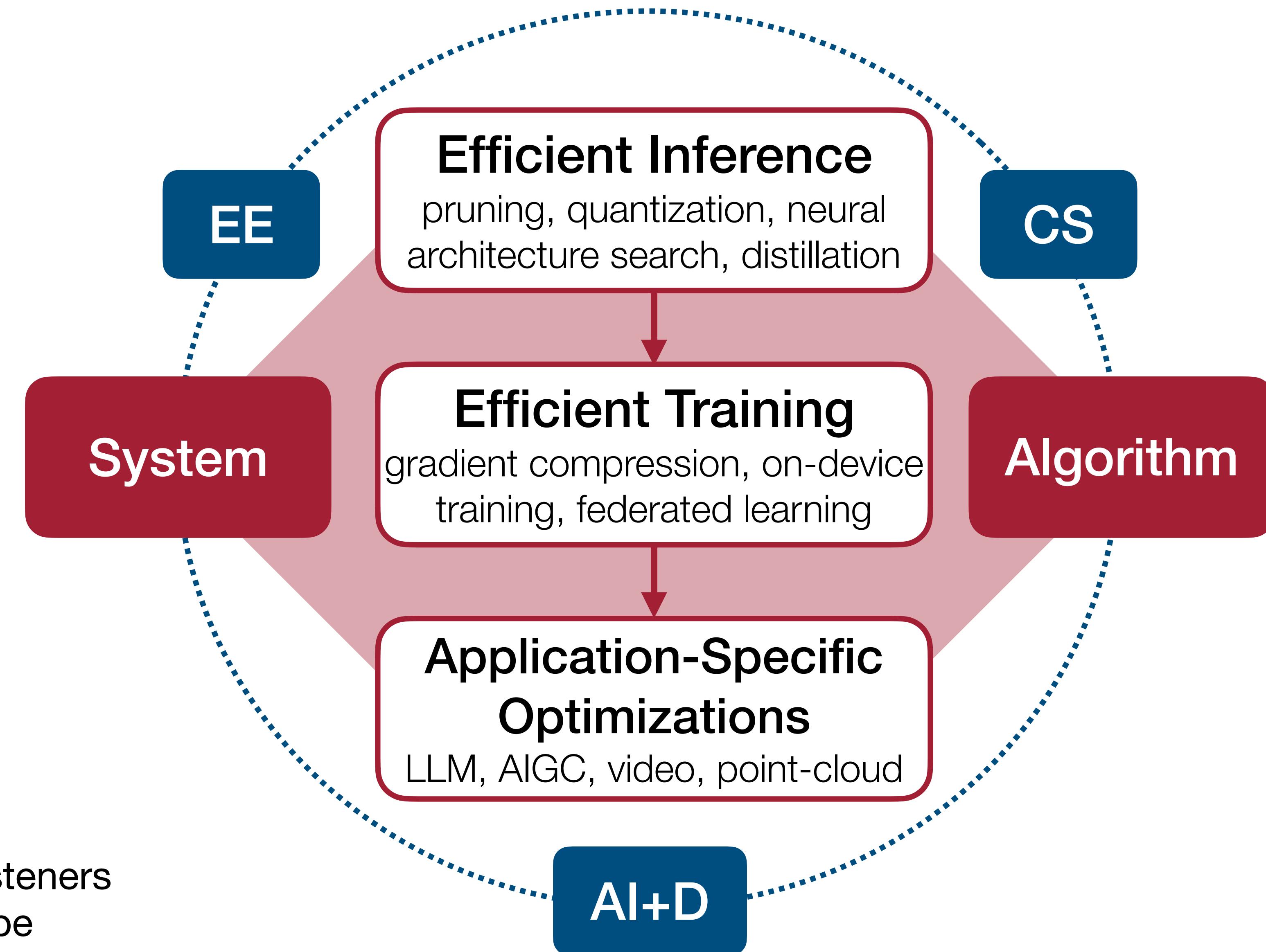
0 forks

### Releases

No releases published  
[Create a new release](#)

### Packages

No packages published



97 registered + 40 listeners

89K views on YouTube

700 students in the open study group on Discord

# EfficientML.ai Course

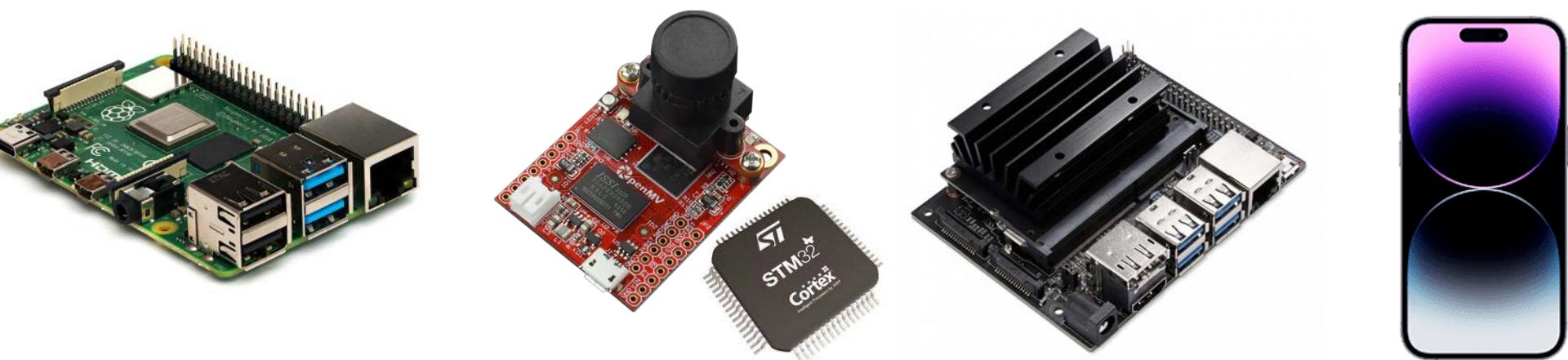
## TinyML and Efficient Deep Learning Computing

6.5940 • Fall 2023 • MIT

Large generative models (e.g., large language models, diffusion models) have shown remarkable performance, but they require a massive amount of computational resources. To make them more accessible, it is crucial to improve their efficiency.

This course will introduce efficient AI computing techniques that enable powerful deep learning applications on resource-constrained devices. Topics include model compression, pruning, quantization, neural architecture search, distributed training, data/model parallelism, gradient compression, and on-device fine-tuning. It also introduces application-specific acceleration techniques for large language models, diffusion models, video recognition, and point cloud. This course will also cover topics about quantum machine learning. Students will get hands-on experience deploying large language models (e.g., LLaMA 2) on a laptop.

- **Time:** Tuesday/Thursday 3:30-5:00 pm Eastern Time
- **Location:** [36-156](#)
- **Office Hour:** Thursday 5:00-6:00 pm Eastern Time, 38-344 Meeting Room
- **Discussion:** [Discord](#)
- **Homework submission:** [Canvas](#)
- **Online lectures:** The lectures will be streamed on [YouTube](#).
- **Resources:** [MIT HAN Lab](#), [HAN Lab Github](#), [TinyML](#), [MCUNet](#), [OFA](#), [SmoothQuant](#)
- **Contact:**
  - Students can ask all course-related questions on [Discord](#).
  - For external inquiries, personal matters, or emergencies, you can email us at [efficientml-staff@mit.edu](mailto:efficientml-staff@mit.edu).
  - If you are interested in getting updates, please sign up [here](#) to join our mailing list to get notified!



This is honestly one of the best set up courses I've taken at MIT

I really like how structured the labs are, and being able to see actual implementations of the techniques we learn about.

I managed the weekly labs and lectures by only watching the course on YouTube. As a researcher, I gained some valuable knowledge from your course. Excellent slides and teaching and useful labs.

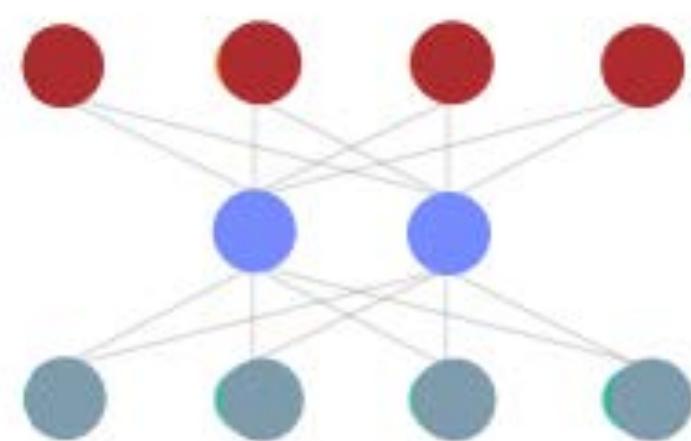
I love how we are using microntroller and focusing on application instead of just theories.

I like the class and I have been able to follow the class easily (which had rarely happened to me in my previous courses)

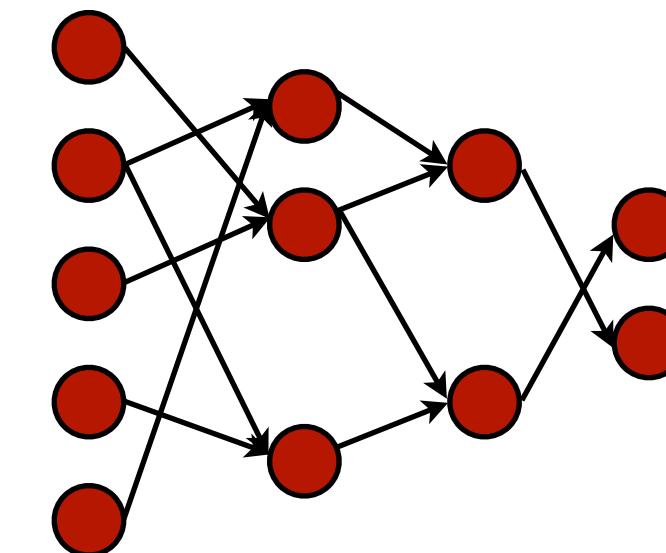
# EfficientML.ai Hands-on Labs

**P Y TORCH**

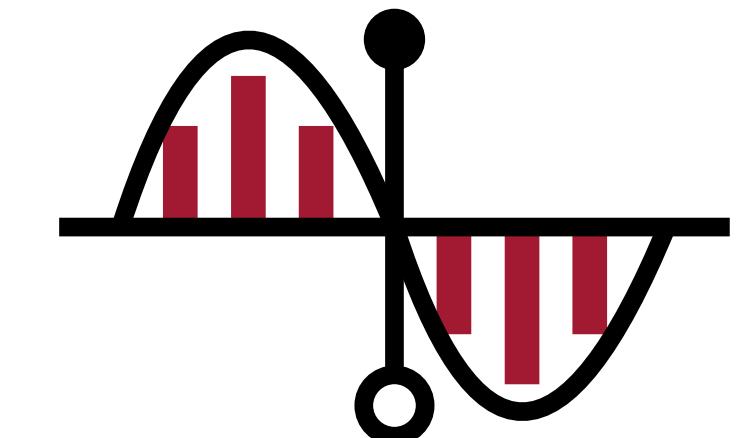
**Lab 0:** Tutorial  
on PyTorch



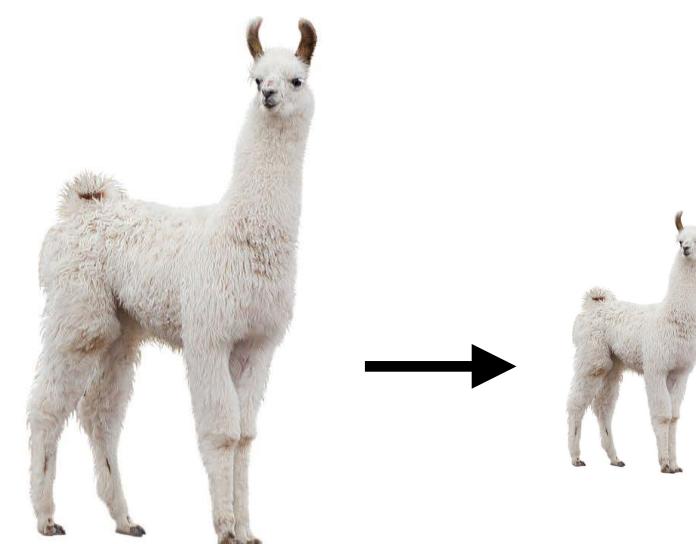
**Lab 3:**  
Neural Architecture  
Search



**Lab 1:**  
Pruning



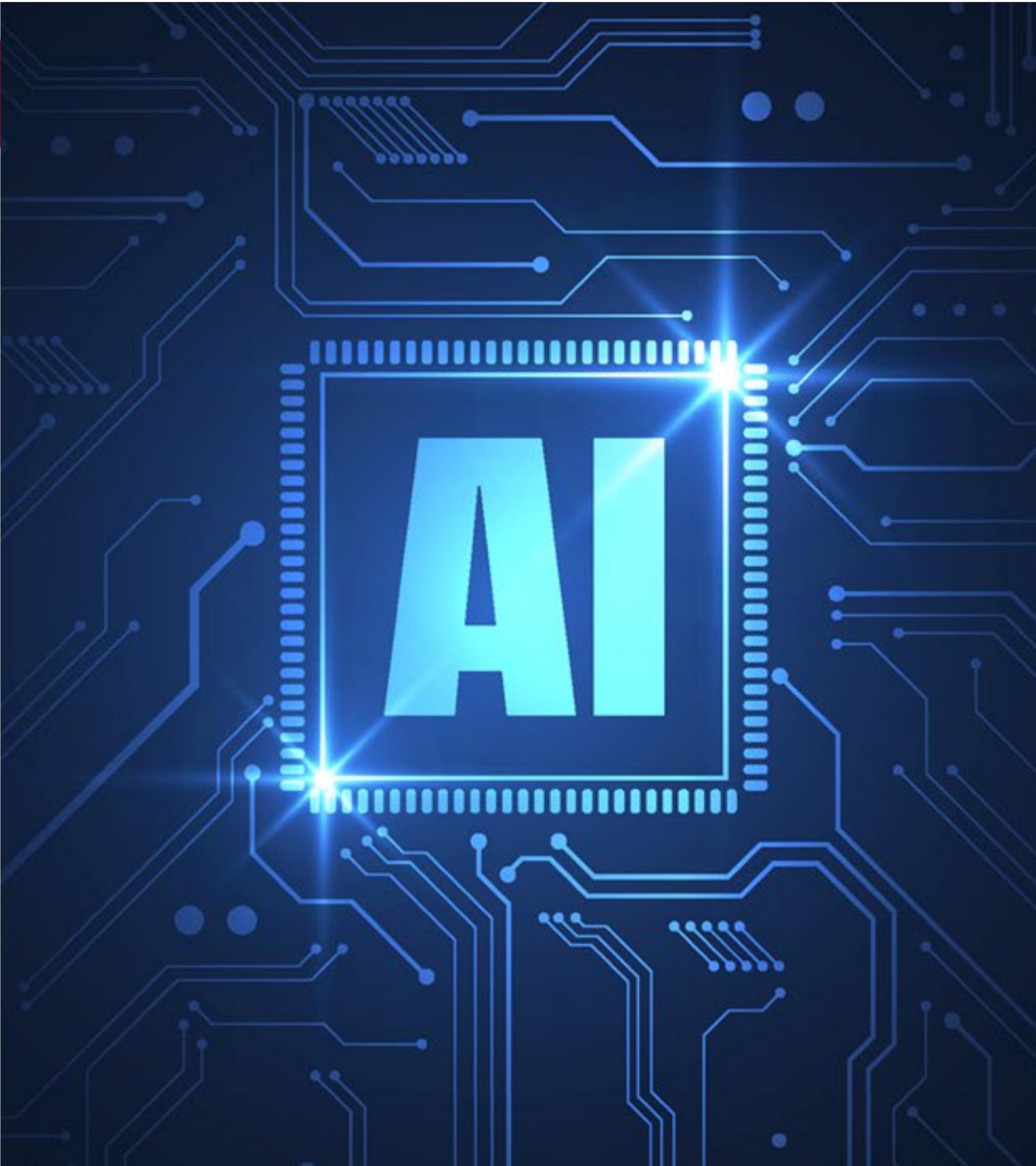
**Lab 2:**  
Quantization



**Lab 4:**  
LLM Compression



**Lab 5:**  
On-Device LLM  
Deployment



# MIT AI Hardware Program

MIT Microsystems Technology Laboratories (SoE)

MIT Quest for Intelligence – Corporate (SCC)

Co-Leads: Jesús del Alamo and Aude Oliva

Internal Advisory Board Chair: Anantha Chandrakasan

# TinyML and Efficient AI Computing



[github.com/mit-han-lab](https://github.com/mit-han-lab)



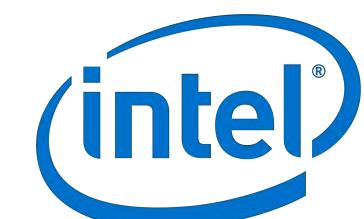
[youtube.com/c/MITHANLab](https://youtube.com/c/MITHANLab)



[songhan.mit.edu](http://songhan.mit.edu)  
[tinyml.mit.edu](http://tinyml.mit.edu)



## Sponsors:



maxim  
integrated



SONY



## Media:

MIT  
Technology  
Review

IEEE  
SPECTRUM

WIRED

engadget

MIT News  
ON CAMPUS AND AROUND THE WORLD

VentureBeat